What's new in Flutter 3.13 - Flutter - Medium

Kevin Chisholm

2D scrolling, faster graphics, Material 3 updates and more

Welcome back to our quarterly Flutter stable release, this time for Flutter 3.13! In just the three months since our last release, we have had 724 pull requests merged and 55 community members authoring their first commit to Flutter! Keep reading to learn about all the new additions and improvements the Flutter community has contributed to this latest release!

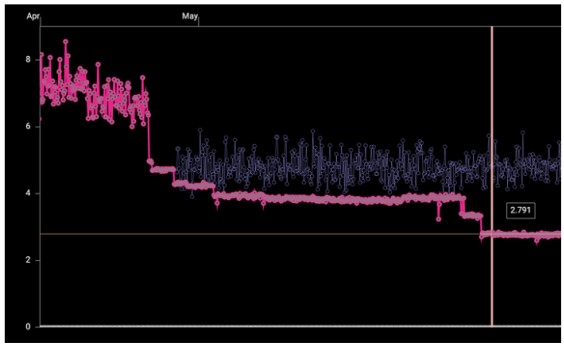
Engine

We've made several improvements to Impeller — our new graphics renderer — and added new Engine APIs for foldable devices.

Impeller

iOS performance improvements

Thanks to the high-quality feedback from Flutter users, in this release we have continued to improve the performance of Impeller on iOS. As a result of many different optimizations, the Impeller renderer on iOS now not only has lower latency (by completely eliminating shader compilation jank), but on some benchmarks also have higher average throughput. In particular, on our flutter/gallery transitions performance benchmark, average frame rasterization time is now around half of what it was with Skia.



Improvements to average frame rasterization time in the Flutter Gallery transitions performance benchmark on an iPhone 11. The time period covered is roughly the time from the 3.10 branch cut to the 3.13 branch cut.

This progress was thanks to these and other optimizations, including:

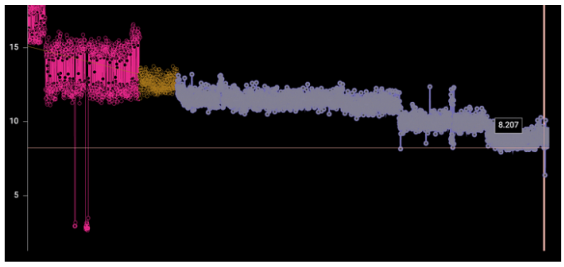
Enabled dirty region management and partial repaint (flutter/engine#40959)
Implemented concurrent render pass encoding (flutter/engine#42028)
Made numerous improvements to text rendering (flutter/engine#41290, flutter/engine#41780, flutter/engine#42417)
Added a fast path for convex shapes to avoid expensive tessellation calls (flutter/engine#41834)
Started to use compute shaders for a few operations (flutter/engine#42192)
More eager culling of out-of-bounds draw operations (flutter/engine#41606)

Fidelity improvements

In 3.10 we announced that wide gamut colors were available under a flag when using Impeller. After hearing and addressing feedback from users, wide gamut colors are now the default on iOS when using Impeller.

Progress update on Impeller on Android

We continue to make progress on the Vulkan backend for Impeller, however it hasn't yet reached the level of quality where an official preview period would be useful. We want to ensure that our users' first experience with Impeller on Android is high quality and we are not quite there yet. We hope to enter a preview period for Impeller on Android in a stable release later this year. Even though Impeller on Android isn't quite ready for preview yet, the OpenGL and Vulkan backends have benefited from many of the backend agnostic optimizations that we've made to Impeller's HAL during the past year. In particular, average frame rasterization times for Android have also improved significantly on the flutter/gallery transitions performance benchmark. Further improvements are in progress so that the preview on Android can be high quality.



Once again, our progress was greatly accelerated by contributions from the community, in particular GitHub user ColdPaleLight, who authored several much appreciated Impeller-related patches, improving fidelity and performance, including adding support for conical gradients.

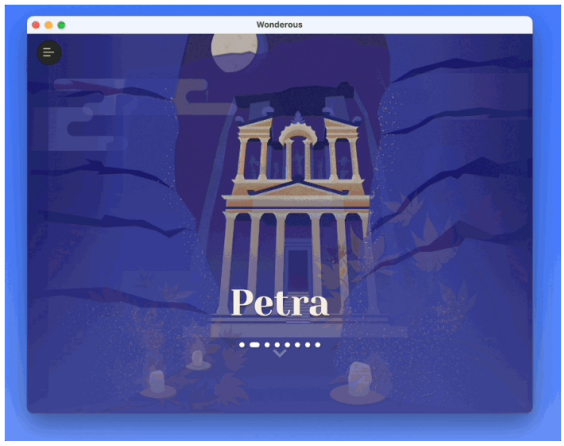
Please continue to follow along with our progress on Impeller using the Impeller project dashboard on GitHub. We greatly appreciate all the feedback and encourage users to continue filing fidelity and performance issues in the issue tracker.

Impeller (and Wonderous) on macOS

In our last stable release, we announced that Impeller, a rewrite of our rendering engine, would be turned on by default for iOS. Since then, we've heard great feedback from customers. Now, we're excited to announce that Impeller for macOS is available in preview. You can test Impeller and enable it in your app by following the guidance on the Impeller page.

We're eager for you to test this out and provide feedback. The best way to help us improve Impeller for macOS is to establish baseline metrics by running your macOS app without SkSL warmup and use DevTools to find instances of jank due to shader compilation. Next, test your app using Impeller — click through and check for bugs, performance improvements or performance regressions. If you notice any issues, we strongly encourage you to file them on GitHub. Be sure to include information about the device you're running on, video recordings, and an export of your performance trace.

Looking to try Impeller on macOS? Install Wonderous from the Mac App Store!



New engine API

Improved foldable support

In order to better support foldable devices, we have added a new API to retrieve various properties of a display. The new getter FlutterView.display returns a Display-class.html" rel="noopener ugc nofollow" target="_blank">Display object. The Display object reports the physical size, the device pixel ratio, and the refresh rate of the display. Check out setPreferredOrientations.html" rel="noopener ugc nofollow" target="_blank">setPreferredOrientations for an example that uses the new API.

Framework

Material

We've made a number of improvements to the Material Framework to 1) offer more platform adaptability, 2) allow for more customization, and 3) add new capabilities.

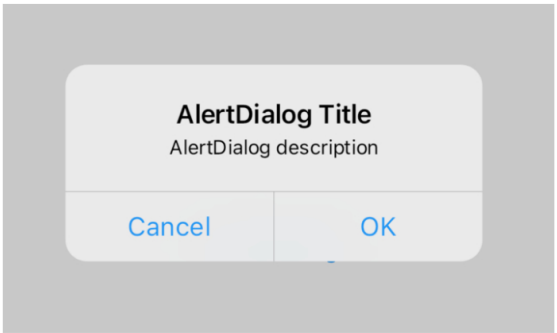
Character recognition in TextField

When using TextField on iOS, users will automatically see an option to use the device camera to recognize characters and insert them into the field.

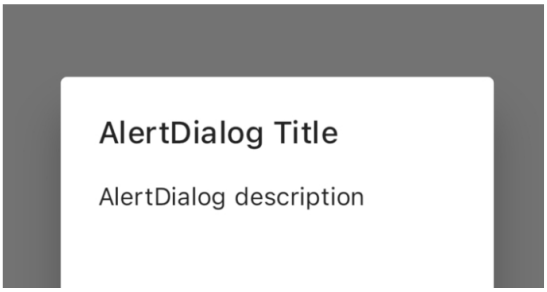
This feature would not be possible without the contributions of community members luckysmg (Author) and tgucio" rel="noopener ugc nofollow" target="_blank">tgucio (reviewer). This feature was a 1000 line and 70 commit effort that bridged the engine and framework! Thank you!

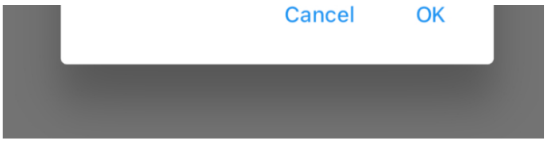
Platform adaptive dialog

An adaptive constructor has been added to the AlertDialog, along with the adaptive function showAdaptiveDialog, to display either a Material or Cupertino dialog depending on the current platform.
Now using AlertDialog.adaptive() uses the CupertinoAlertDialog widget on iOS:



And Material AlertDialog on Android.





CupertinoDatePicker with month and year

Adds a monthYear mode to the CupertinoDatePicker.

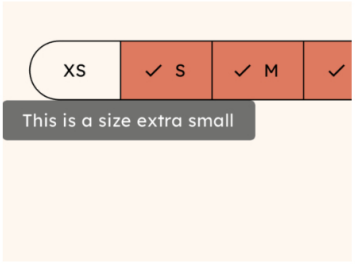
Cupertino (iOS-style) check styled radio

The useCheckmarkStyle property has been added to CupertinoRadio. This also allows the Radio.adaptive and RadioListTile.adaptive widgets to control whether they use the checkmark style on iOS.

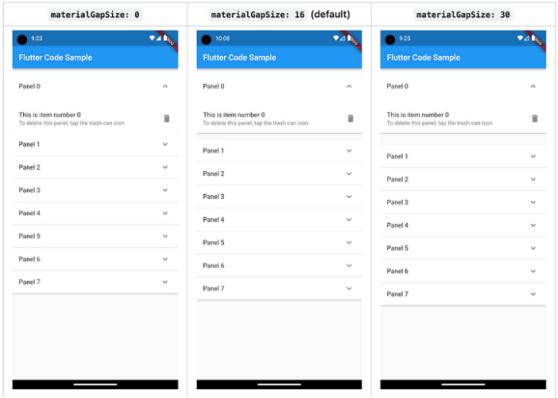
More customization options for Material widgets

There have been several improvements that make it easier to customize the design of the Material widgets:
You can now use the error property for InputDecoration (as opposed to a string) to customize the error widget that is shown on text fields:

You can now add tooltips to ButtonSegment:

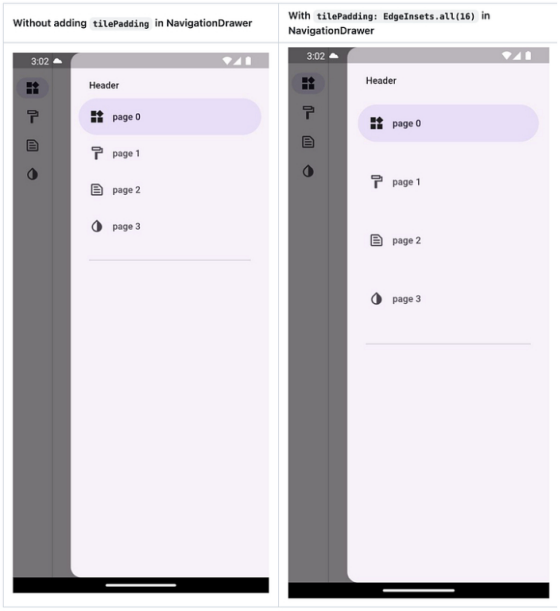


You can now customize the gap in ExpansionPanelList using the materialGapSize property

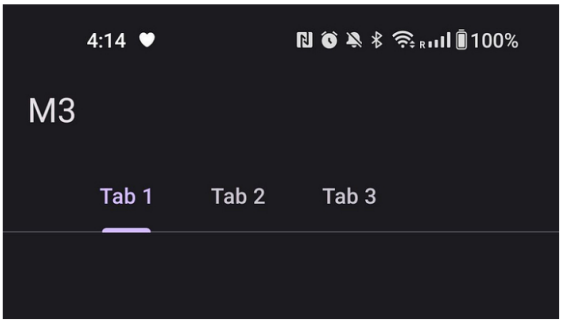


You can now customize the trackOutlineWidth for Switch

You can now customize the padding with the tilePadding property on NavigationDrawer

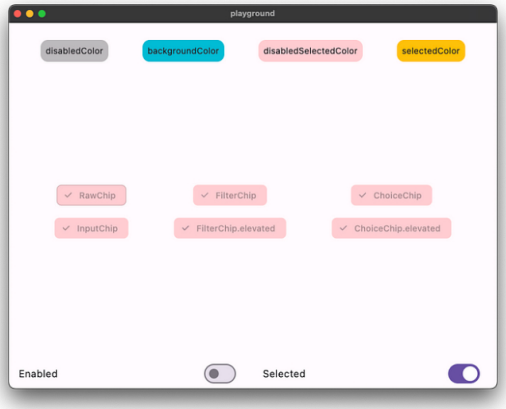


You can choose how to align the tabs using the alignment property for TabBar



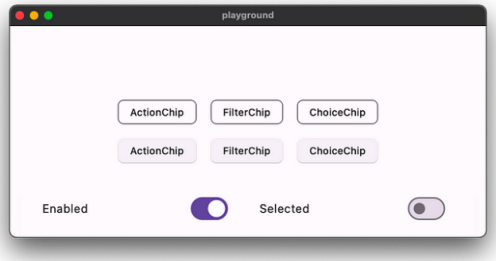
MaterialState color for chips

This makes it possible to customize the color of the chips in all of the different states.



Elevated Chips

FilterChip.elevated, ChoiceChip.elevated, and ActionChip.elevated variants have been added in accordance with the Material 3 specs.



onSubmitted to SearchBar

Allows for a different action to be initiated when a user finishes the text entry

and presses the Done button on the keyboard.

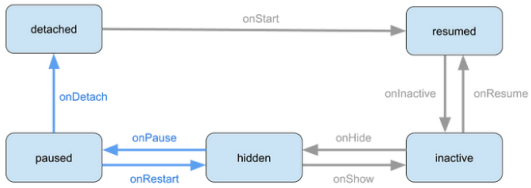
BaseTapAndDragGestureRecognizer

A base class has been added for a family of classes, which includes support for tap + pan (TapAndPanGestureRecognizer), and tap + horizontal drag (TapAndHorizontalDragGestureRecognizer). These classes have already been used to implement native text field gestures. However, they're also great for other use cases — for example, scaling a widget using a double tap + vertical drag gesture.

App Lifecycle Changes

AppLifeCycleListener

AppLifecycleListener class was added for listening to changes in the application lifecycle, and responding to requests to exit the application.



Scrolling

TwoDimensional scrolling foundation

This release of Flutter also contains the foundation for building widgets that scroll in two dimensions, which means a bunch of new classes to build with, including:

ChildVicinity, a representation similar to an index in a one dimensional scrollview, representing the relative position of children in two dimensions.

TwoDimensionalChildDelegate, similar to SliverChildDelegate with equally similar subclasses: TwoDimensionalChildBuilderDelegate & TwoDimensionalChildListDelegate

TwoDimensionalScrollView, an abstract base class that creates a TwoDimensionalScrollable and TwoDimensionalViewport, matching the same model as the one dimensional ScrollView.

RenderTwoDimensionalViewport, and finally, the workhorse of laying out box children in two dimensions.

Scrolling in two dimensions also comes with some new interactions, including diagonal scrolling. See DiagonalDragBehavior for new interaction types, and configure them on your TwoDimensionalScrollView or TwoDimensionalScrollable.

We conducted a user study in order to develop this foundation for developers to be able to build whatever they could imagine while scrolling in all directions. Check out an example of a simple, lazy loading, two dimensional grid implemented in this DartPad in about 200 lines of code!

The Flutter team is already at work building two dimensional scrolling widgets on top of this framework, coming soon in the two_dimensional_scrollables package.

New slivers

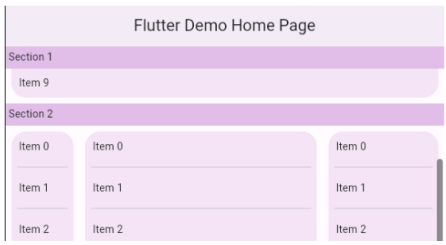
Flutter 3.13 brings with it a new set of slivers for composing unique scrolling effects.

SliverMainAxisGroup and SliverCrossAxisGroup-class.html" rel="noopener ugc nofollow" target="_blank">SliverCrossAxisGroup both support arranging multiple slivers together. In the main axis, one effect this can create is sticky headers, allowing pinned headers to be pushed out of view as each group of slivers scrolls by.

The cross axis grouping allows for slivers to be arranged side by side in the viewport, with (also new) widgets like SliverCrossAxisExpanded and SliverConstrainedCrossAxis-class.html" rel="noopener ugc nofollow" target="_blank">SliverConstrainedCrossAxis capable of determining the allotment of space for each grouped sliver in the cross axis.

Also new to the sliver library is DecoratedSliver, similar to DecoratedBox-class.html" rel="noopener ugc nofollow" target="_blank">DecoratedBox. This allows users to embellish a sliver, which could even be a sliver group, with a Decoration-class.html" rel="noopener ugc nofollow" target="_blank">Decoration.

See all of these new slivers in action in this DartPad example.



Accessibility

Accessibility updates

The onOffSwitchLabels accessibility property was added for CupertinoSwitch to display I/O labels

The FocusSemanticEvent has been added. However, it should be used with caution as it might break a users' expectation of how a11y focus works. IconButton's isSelected is now available to screen readers.

Platforms

Android

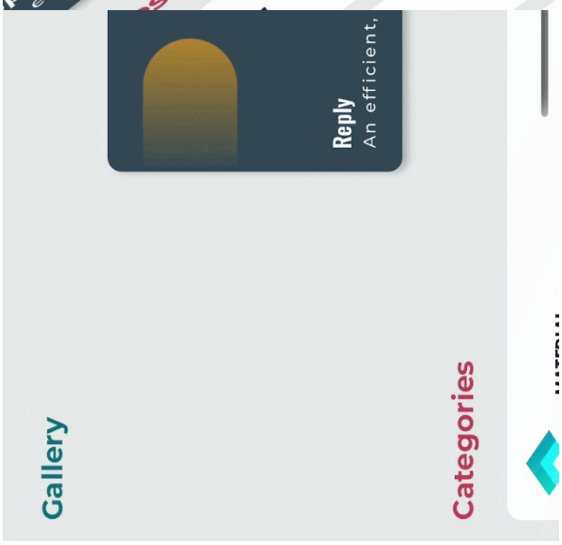
New support targets

With this release, Flutter now supports targeting Android 14/ API 34. While we are still working on a few new features in Android 14 (i.e. predictive back navigation), we have thoroughly tested this release against the new Android SDK and prioritize fixing any related issues you may find.

iOS

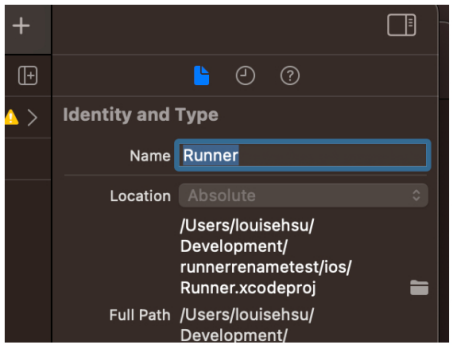
Reduced rotation distortion on iOS

When an iOS screen rotates, Flutter apps would previously experience some distortion that looked different from native iOS applications. We've made some modifications to reduce the distortion:



Renaming Runner

When a Flutter iOS app is created, a *Runner* Xcode project and Xcode workspace are created in the `/ios` folder. Now, you can rename the workspace or project so that you don't end up with a list of *Runners*.



Preparing for iOS 17 and Xcode 15

With the impending release of iOS 17 and Xcode 15, users who desire to develop using this toolchain will need to be on Flutter 3.13. In addition, when downloading Xcode 15, make sure you also download the iOS 17 simulator.

Games

Flutter games updates

We launched the Flutter casual games toolkit in 2022 with a game template, tutorials, documentation, community spaces, and GCP/Firebase/Ad credits to jumpstart game development for Flutter Developers. Since then, tens of thousands of games have been published using Flutter! Since launch, we've actively engaged and surveyed Flutter game developers to find out how we could improve the games toolkit. Almost all of them mentioned wanting more resources and sample code to help them better design, develop and monetize their games.

Today, we are releasing a new update to the Flutter Games web page with a carousel of video resources and new games to learn about while building in Flutter. We have a number of new updates to the toolkit coming in the next few months with additional resources and samples to kickstart your game development journey.

As a first step, we partnered with AdMob in July 2023 and co-hosted an exclusive UX design and Monetization workshop dedicated to Flutter game devs. ~ 100 developers joined us via a live interactive webinar, and gave the session a 4.6/5.0 satisfaction rating. We hope to summarize the workshop content and share these insights more broadly with all of you soon.

We are actively working on more updates, so please stay tuned! If you are already using the games toolkit, and would love to send us ideas for future improvement, please don't hesitate to email us at flutter-games@google.com.

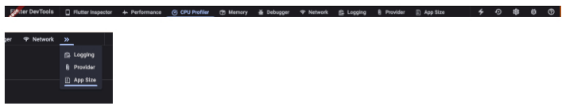
Tooling

DevTools

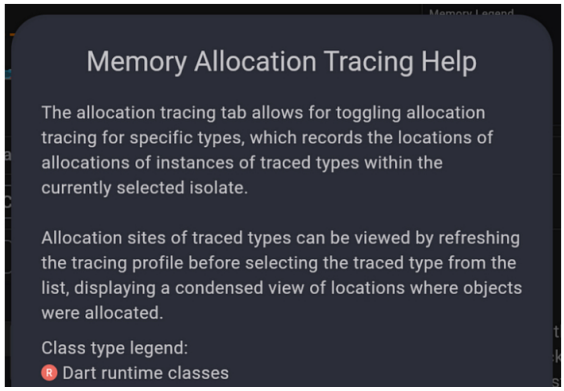
New DevTools features

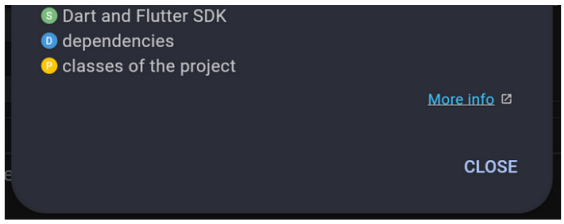
We've made improvements to the performance and usability of DevTools that include:

We added a new overflow menu on the navigation bar to handle cases when the list of tabs can't be displayed at once.



We added a legend for class types on the Memory tab.





Additionally, we made scrolling a tree table in the CPU profiler faster and smoother. In the debugger, we've made searching in a file, or searching for a file up to 5x faster.

To learn more, check out the release notes for DevTools 2.25.0, and DevTools 2.24.0.

Breaking changes and deprecations

Breaking changes

Material 3 by default in the next release

We're excited to announce that in the **next** Flutter stable release we plan to change the ThemeData useMaterial3 default from false to true. In other words, applications will get the Material 3 colors, text styles, and other visuals, by default.

The Material 3 demo should be helpful for previewing the differences between M2 and M3.

Android supported platforms

Flutter no longer supports the Android Jelly Bean API levels (16, 17, and 18). The good news is that most apps should be migrated to this new minSdkVersion by default.

However, if you were not migrated automatically, it could be because you made changes to your module level build.gradle, and you might need to increase the minSdkVersion manually. To update, locate the module level `build.gradle` from the root of your Flutter project. It is typically found at <`YOUR PROJECT`>`/android/app/build.gradle`. Bump `minSdkVersion` version to 19. If you see flutter.minSdkVersion and it's at least 19, then your minimum is set correctly.

Flutter plugins won't be migrated by default, so plugin authors should update the `minSdkVersion` in the top level `build.gradle` file found at <`YOUR PLUGIN`>`/android/build.gradle`.

List of changes and migration guides

Breaking changes in this release include deprecated APIs that expired after the release of v3.10. To see all affected APIs, along with additional context and migration guidance, see the deprecation guide for this release. Many of these are supported by Flutter Fix, including quick fixes in the IDE, and bulk apply with the dart fix command.

As always, many thanks to the community for contributing tests, they help us identify these breaking changes. To learn more, check out Flutter's breaking change policy.

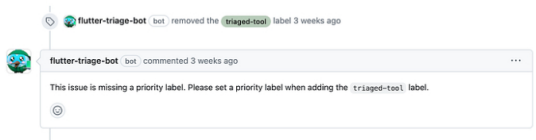
Contributions

Flutter repository priorities and triage

Triage updates

Over the past few months we have adopted a new set of definitions for our priorities (P0-P3). This brings us more in line with the definitions used by most other open source projects, and simplifies the decisions we have to make regarding how important bugs are — rather than 7 categories, we now only have 4. Hopefully this will also help us communicate more effectively with those of you who file bugs and then wonder when things will be fixed!

We've also introduced a new triage scheme for our teams, which we hope will make it harder for issues to fall between the cracks. If you are active in our issue database you may see our new bot commenting and adding or removing labels:



This bot also integrates with our Discord, helping us keep on top of what is happening on GitHub. We hope it will make us more productive in the long run, but bear with us while we get used to the new system!

Conclusion

As we come to the end of this announcement, I want to acknowledge that we would not be where we are today without the efforts of our fantastic community!

For a full list of PRs that were included in this release, please check out the release notes and change log section for this release.

Flutter 3.13 is available in stable today, and includes Dart 3.1. To get started

Flutter 3.13 is available in stable today, and includes Dart 3.1. To get started with these newest updates, all it takes is a `flutter upgrade`.

See you all again soon!