

[Get started](#)

[Samples & tutorials](#)

[Development](#)

▶ [User interface](#)

▶ [Data & backend](#)

▶ [Accessibility & internationalization](#)

▶ [Platform integration](#)

▼ [Packages & plugins](#)

[Using packages](#)

[Developing packages & plugins](#)

[Flutter Favorites program](#)

[Background processes](#)

[Android plugin upgrade](#)

[Package site](#) [↗](#)

▶ [Add Flutter to existing app](#)

▶ [Tools & techniques](#)

▶ [Migration notes](#)

Background processes

[Docs](#) > [Development](#) > [Packages & plugins](#) > [Background processes](#)

Have you ever wanted to execute Dart code in the background—even if your app wasn’t the currently active app? Perhaps you want to implement a process that watches the time, or that catches camera movement. In Flutter, you can execute Dart code in the background.

The mechanism for this feature involves setting up an isolate. *Isolates* are Dart’s model for multithreading, though an isolate differs from a conventional thread in that it doesn’t share memory with the main program. You’ll set up your isolate for background execution using callbacks and a callback dispatcher.

For more information and a geofencing example that uses background execution of Dart code, see the Medium article by Ben Koi [Executing Dart in the Background with Flutter Plugins and Geofencing](#). At the end of this article, you’ll find links to example code, relevant documentation for Dart, iOS, and Android.

