Skooldio

# Deep Dive into Flutter Debugging

ทำความรู้จักกับ DevTools ว่าจะช่วยในการทำงานอย่างไร

# About
# This Section

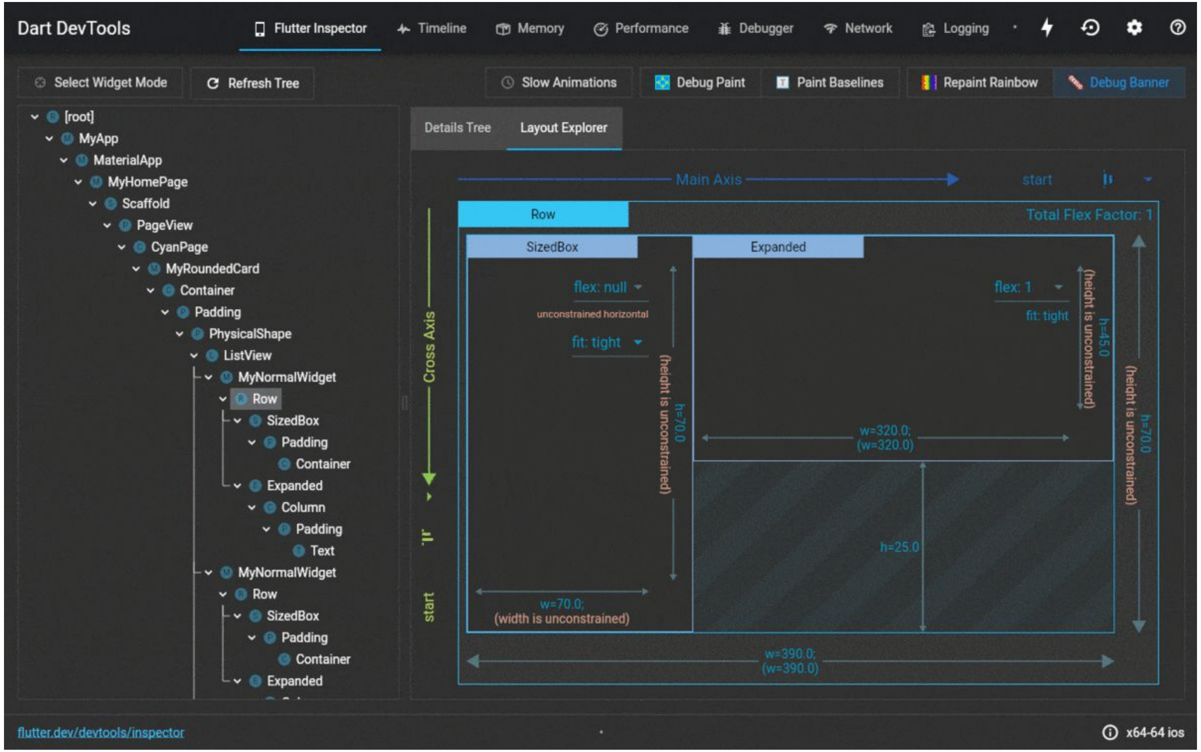
- Flutter Devtools
- App Size Inspection and Optimization
- How to Monitor Flutter Performance
- Speed Up Your Application
  with Pre-Warmed Skia

Skooldio

# Flutter DevTools

# Flutter DevTools

- Inspect the UI Layout and State of a Flutter App.
- CPU Profiling for a Flutter or Dart App.
- Network Profiling for a Flutter App.
- Debug Memory Issues in a Flutter or Dart Command-Line App.
- Analyze Code and App Size.



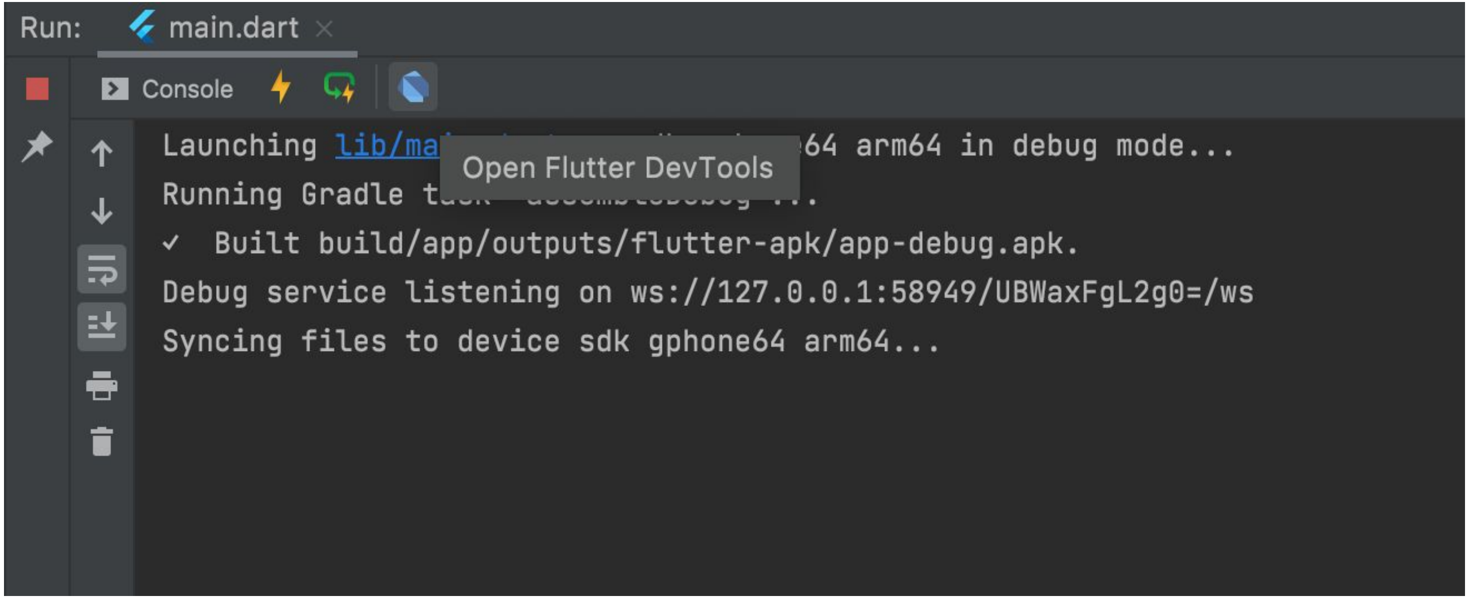
# DevTools Features

- Inspect the UI Layout and State of a Flutter App.
- CPU Profiling for a Flutter or Dart App.
- Network Profiling for a Flutter App.
- Debug Memory Issues in a Flutter or Dart Command-Line App.
- Analyze Code and App Size.

# CLI

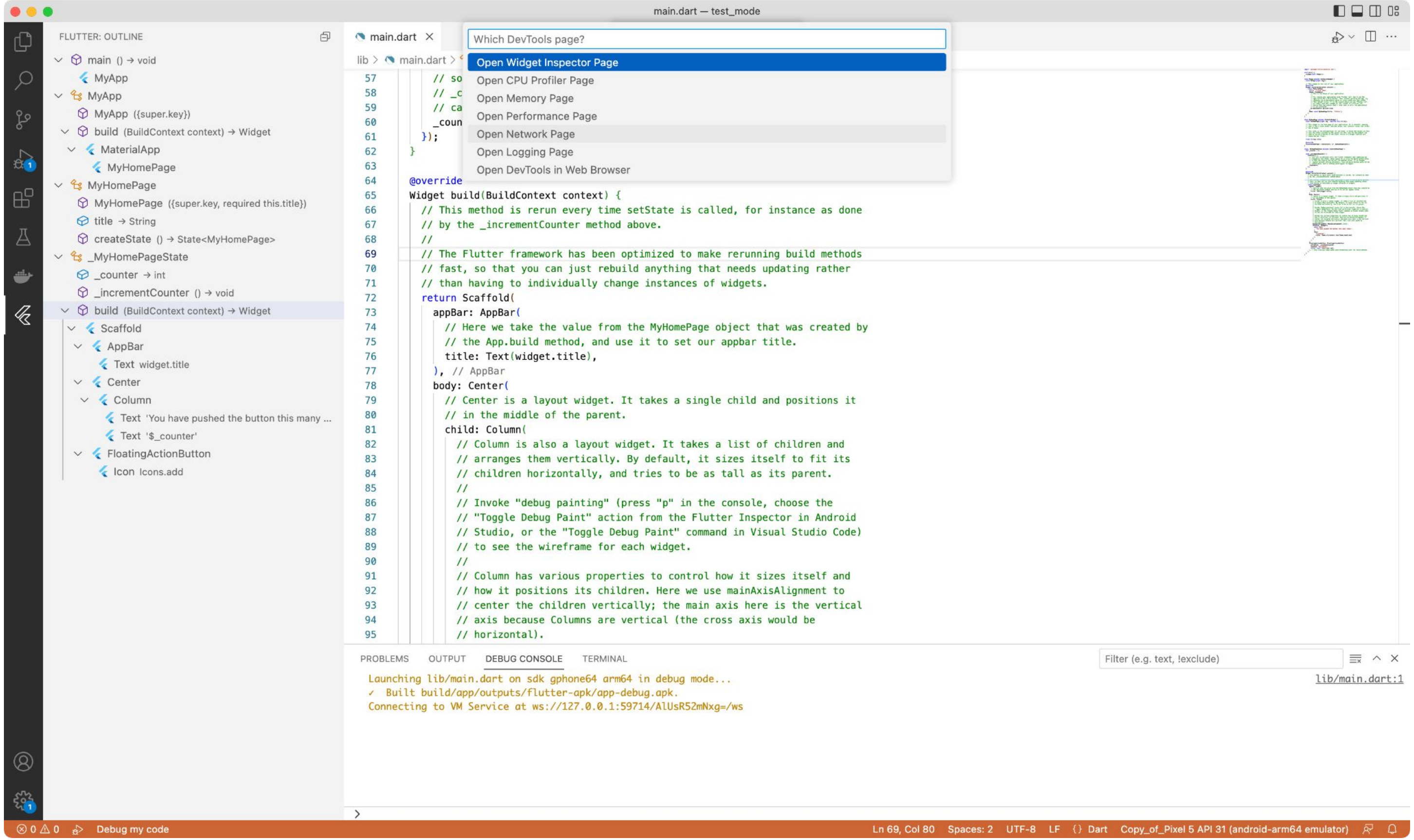
```
An Observatory debugger and profiler on sdk gphone64 arm64 is
available at:
http://127.0.0.1:58652/SXGY6buJmPM=/
The Flutter DevTools debugger and profiler on sdk gphone64 arm64
is available
at: http://127.0.0.1:9101?uri=http://127.0.0.1:58652/SXGY6buJmPM=/
```

# Android Studio



```
Launching lib/ma                   64 arm64 in debug mode...
Running Gradle t                      ...
✓  Built build/app/outputs/flutter-apk/app-debug.apk.
Debug service listening on ws://127.0.0.1:58949/UBWaxFgL2g0=/ws
Syncing files to device sdk gphone64 arm64...
```

# VS Code

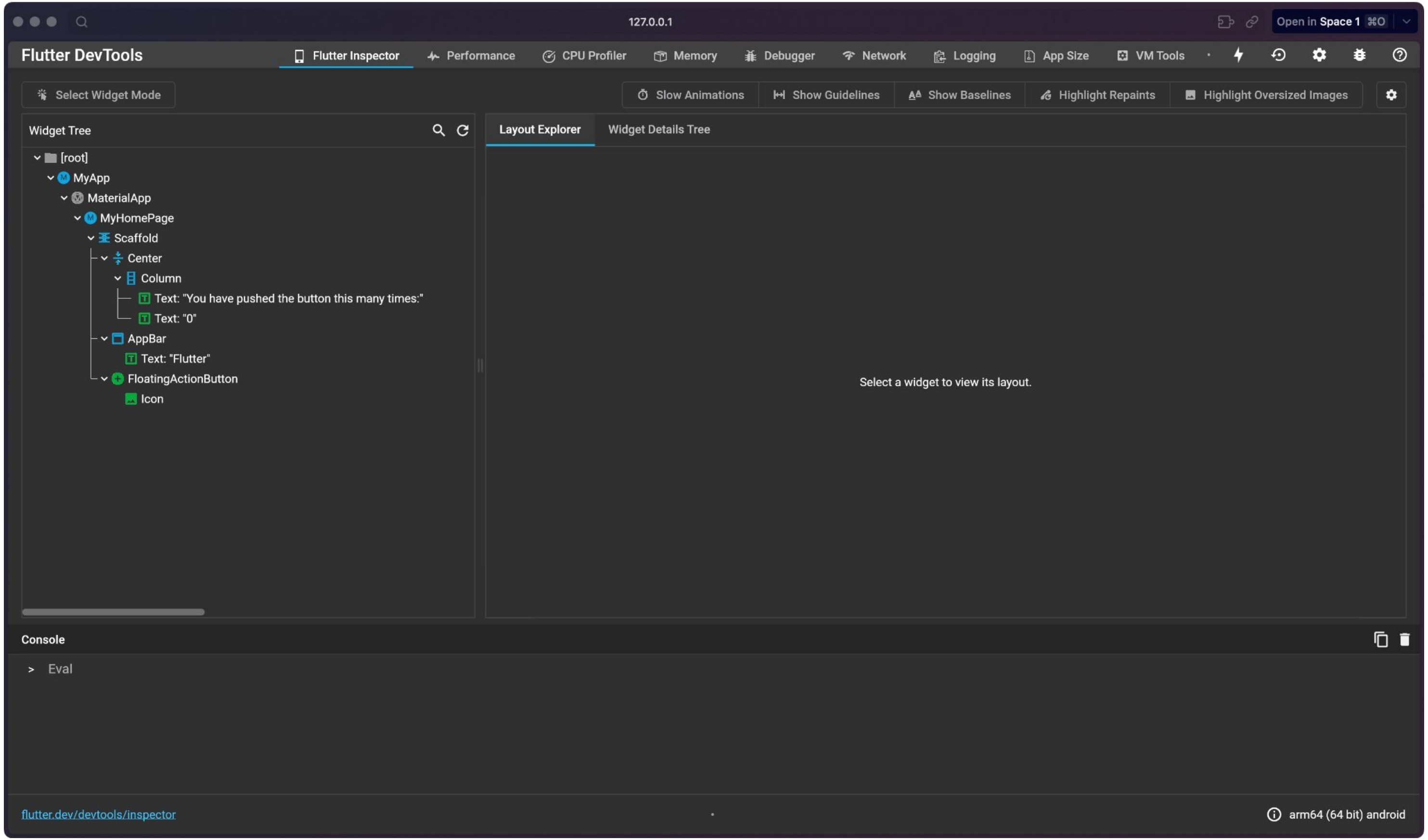


```
      // so
      // _c
      // ca
      _coun
    });
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build methods
    // fast, so that you can just rebuild anything that needs updating rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ), // AppBar
      body: Center(
        // Center is a layout widget. It takes a single child and positions it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children and
          // arranges them vertically. By default, it sizes itself to fit its
          // children horizontally, and tries to be as tall as its parent.
          //
          // Invoke "debug painting" (press "p" in the console, choose the
          // "Toggle Debug Paint" action from the Flutter Inspector in Android
          // Studio, or the "Toggle Debug Paint" command in Visual Studio Code)
          // to see the wireframe for each widget.
          //
          // Column has various properties to control how it sizes itself and
          // how it positions its children. Here we use mainAxisAlignment to
          // center the children vertically; the main axis here is the vertical
          // axis because Columns are vertical (the cross axis would be
          // horizontal).
```

```
Launching lib/main.dart on sdk gphone64 arm64 in debug mode...
✓  Built build/app/outputs/flutter-apk/app-debug.apk.
Connecting to VM Service at ws://127.0.0.1:59714/AlUsR52mNxg=/ws
```

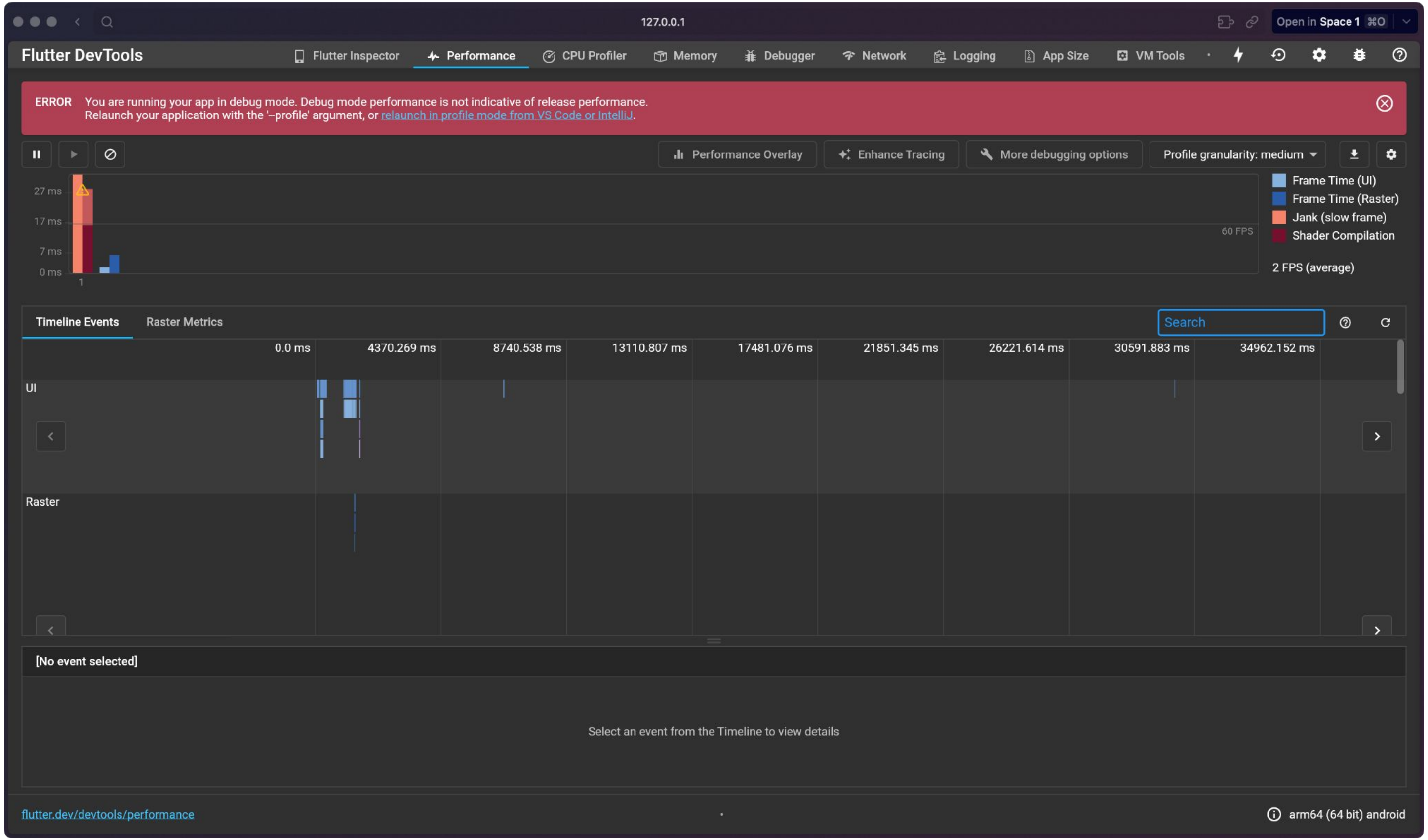
# UI Inspector



```
[root]
  MyApp
    MaterialApp
      MyHomePage
        Scaffold
          Center
            Column
              Text: "You have pushed the button this many times:"
              Text: "0"
          AppBar
            Text: "Flutter"
          FloatingActionButton
            Icon
```

Select a widget to view its layout.

# Performance Inspector



ERROR You are running your app in debug mode. Debug mode performance is not indicative of release performance.
Relaunch your application with the '–profile' argument, or relaunch in profile mode from VS Code or IntelliJ.

Profile granularity: medium

2 FPS (average)

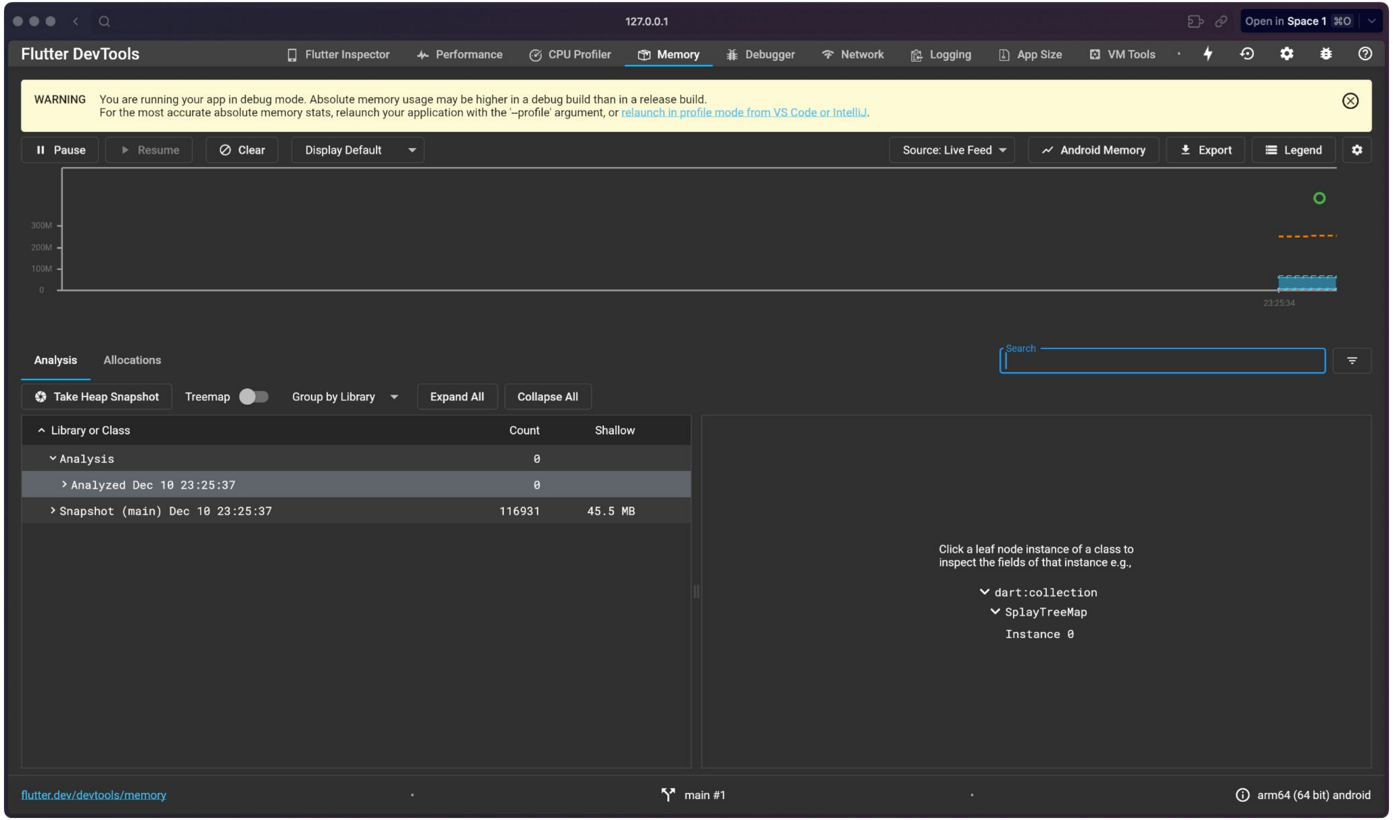
[No event selected]

Select an event from the Timeline to view details

flutter.dev/devtools/performance

arm64 (64 bit) android

# Memory Debugger



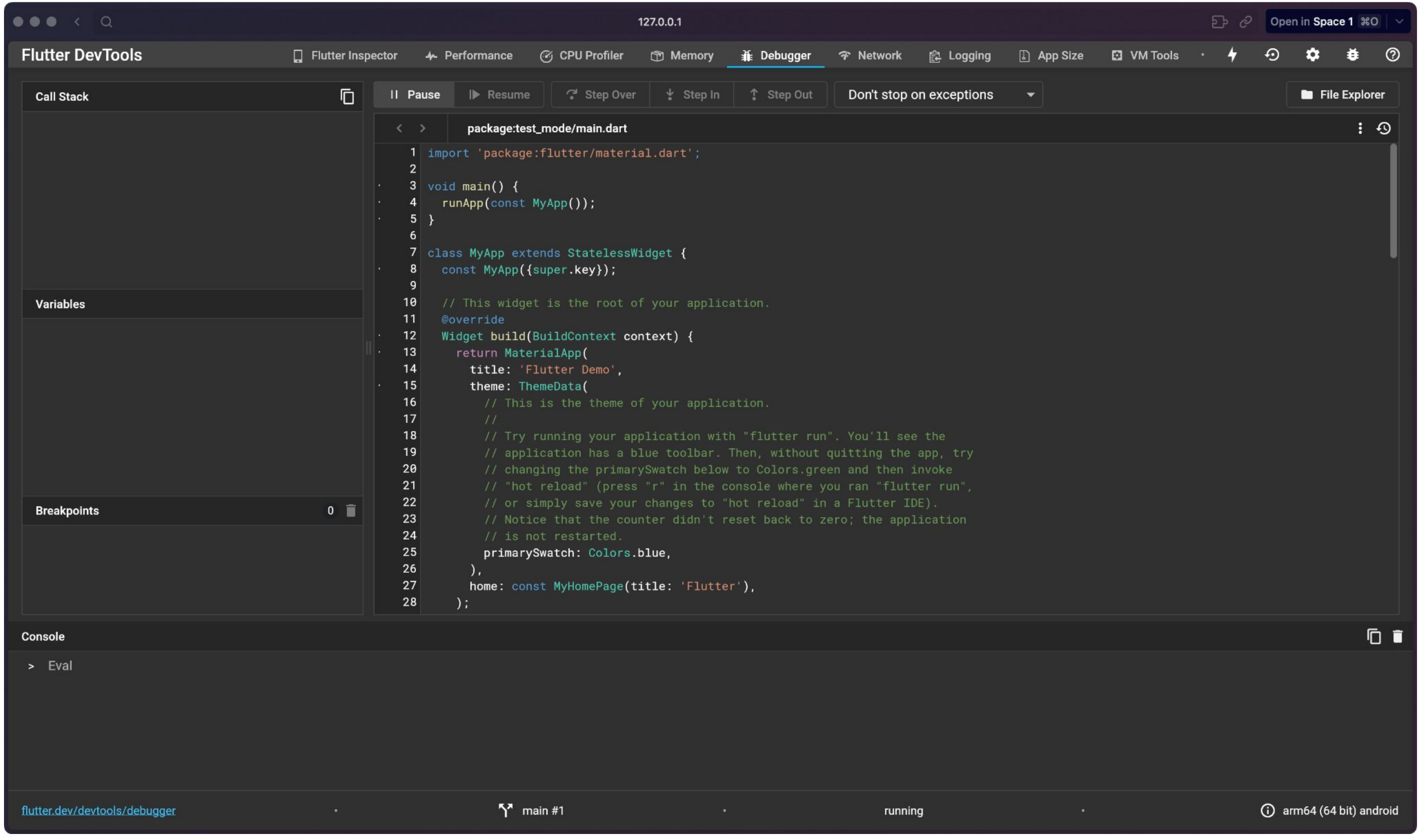
WARNING You are running your app in debug mode. Absolute memory usage may be higher in a debug build than in a release build.
For the most accurate absolute memory stats, relaunch your application with the '--profile' argument, or relaunch in profile mode from VS Code or IntelliJ.

| Library or Class | Count | Shallow |
|---|---|---|
| Analysis | 0 | |
| Analyzed Dec 10 23:25:37 | 0 | |
| Snapshot (main) Dec 10 23:25:37 | 116931 | 45.5 MB |

Click a leaf node instance of a class to inspect the fields of that instance e.g.,

```
dart:collection
  SplayTreeMap
    Instance 0
```

# Source-Code Debugger



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter'),
    );
```

Skooldio

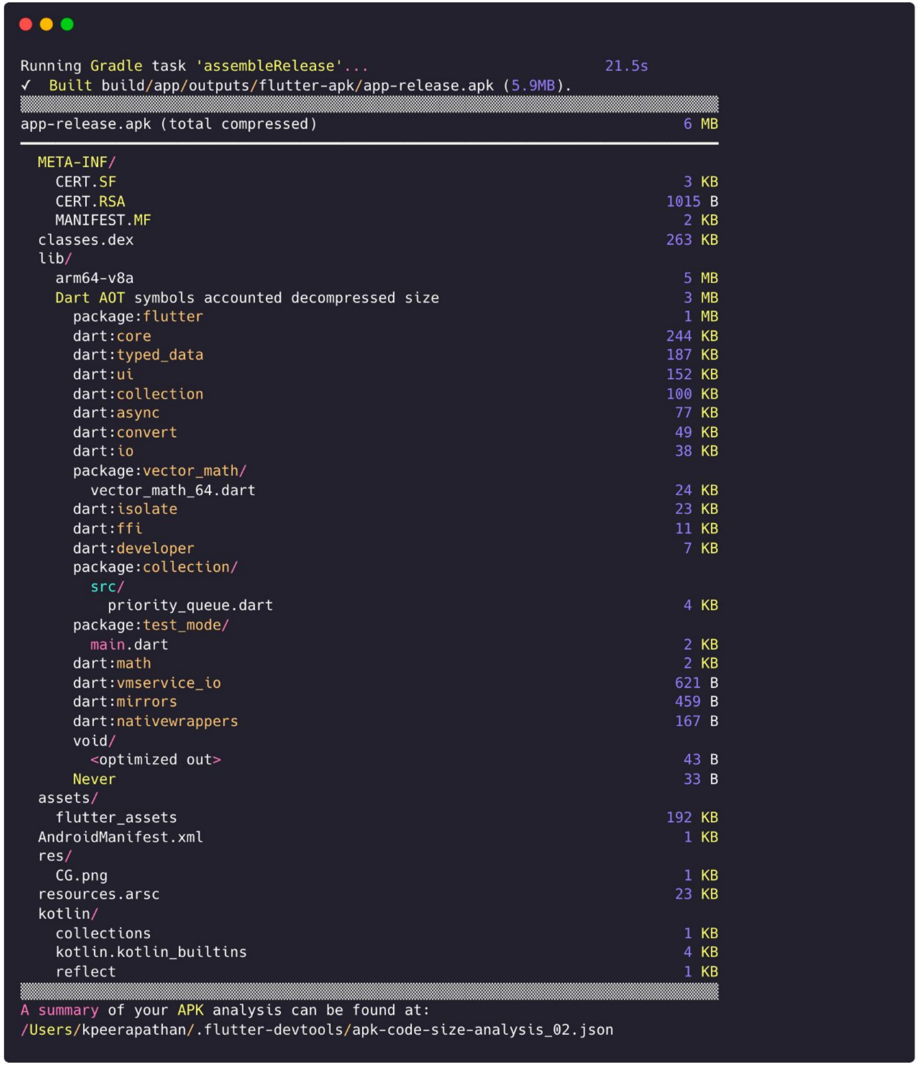
# Demo

Skooldio

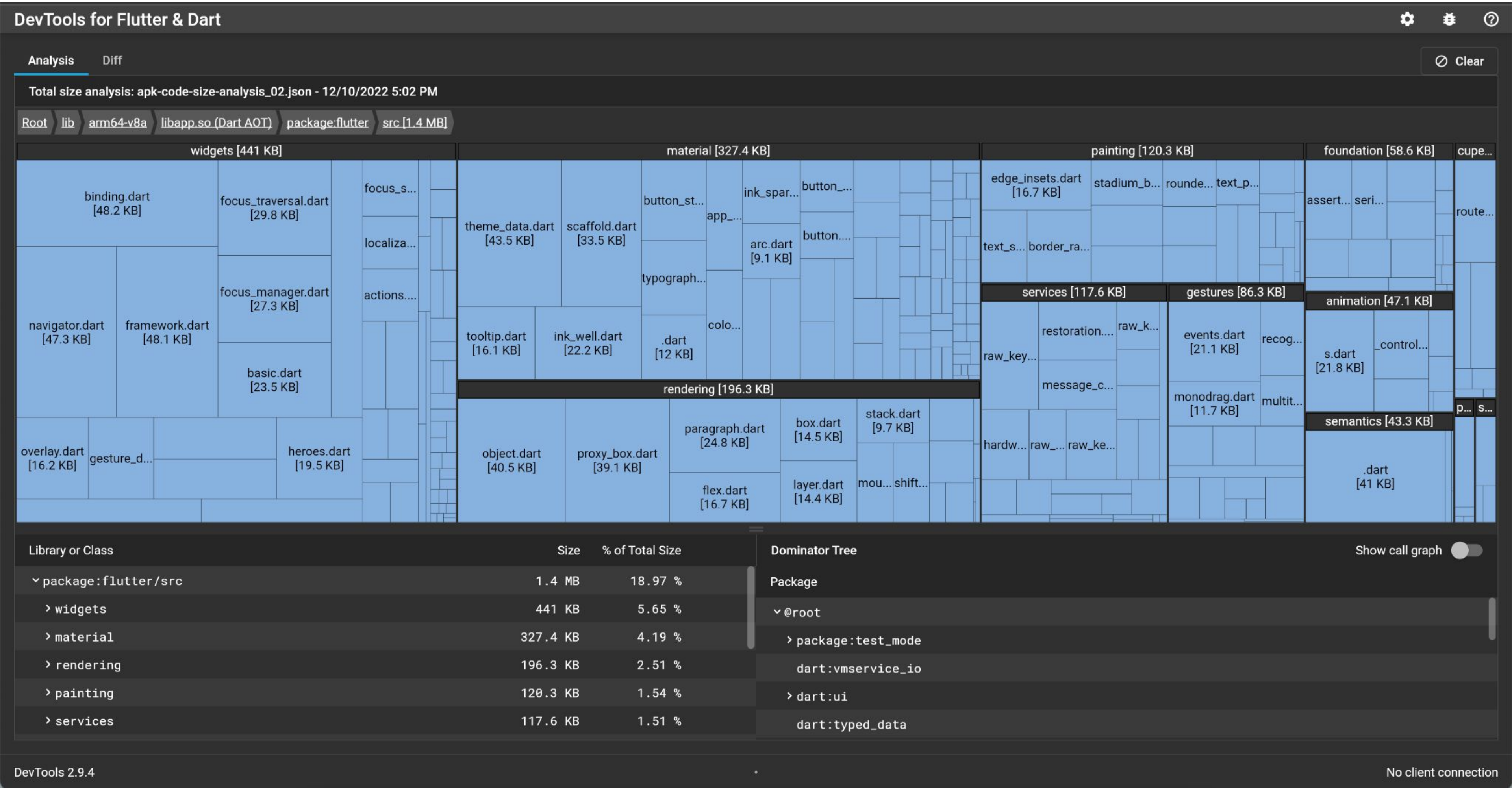
# App Size Analysis

```
flutter build apk --analyze-size
flutter build appbundle --analyze-size
flutter build ios --analyze-size
flutter build linux --analyze-size
flutter build macos --analyze-size
flutter build windows --analyze-size
```

apk-code-size-analysis_02.json



```
Running Gradle task 'assembleRelease'...                           21.5s
✓  Built build/app/outputs/flutter-apk/app-release.apk (5.9MB).
app-release.apk (total compressed)                                        6 MB
  META-INF/
    CERT.SF                                                               3 KB
    CERT.RSA                                                           1015 B
    MANIFEST.MF                                                           2 KB
  classes.dex                                                           263 KB
  lib/
    arm64-v8a                                                             5 MB
    Dart AOT symbols accounted decompressed size                          3 MB
      package:flutter                                                     1 MB
      dart:core                                                         244 KB
      dart:typed_data                                                   187 KB
      dart:ui                                                           152 KB
      dart:collection                                                   100 KB
      dart:async                                                         77 KB
      dart:convert                                                       49 KB
      dart:io                                                            38 KB
      package:vector_math/
        vector_math_64.dart                                              24 KB
      dart:isolate                                                       23 KB
      dart:ffi                                                           11 KB
      dart:developer                                                      7 KB
      package:collection/
        src/
          priority_queue.dart                                             4 KB
      package:test_mode/
        main.dart                                                         2 KB
      dart:math                                                           2 KB
      dart:vmservice_io                                                 621 B
      dart:mirrors                                                      459 B
      dart:nativewrappers                                               167 B
      void/
        <optimized out>                                                  43 B
      Never                                                              33 B
  assets/
    flutter_assets                                                      192 KB
  AndroidManifest.xml                                                     1 KB
  res/
    CG.png                                                                1 KB
  resources.arsc                                                         23 KB
  kotlin/
    collections                                                           1 KB
    kotlin.kotlin_builtins                                                4 KB
    reflect                                                               1 KB
A summary of your APK analysis can be found at:
/Users/kpeerapathan/.flutter-devtools/apk-code-size-analysis_02.json
```



Total size analysis: apk-code-size-analysis_02.json - 12/10/2022 5:02 PM

| Library or Class | Size | % of Total Size |
|---|---|---|
| ⌄ package:flutter/src | 1.4 MB | 18.97 % |
| › widgets | 441 KB | 5.65 % |
| › material | 327.4 KB | 4.19 % |
| › rendering | 196.3 KB | 2.51 % |
| › painting | 120.3 KB | 1.54 % |
| › services | 117.6 KB | 1.51 % |

Dominator Tree

Show call graph

Package

```
⌄ @root
  › package:test_mode
    dart:vmservice_io
  › dart:ui
    dart:typed_data
```

DevTools 2.9.4

No client connection

# Size Optimization

- Remove Unused Resources
- Minimize Resource Imported from Libraries
- Compress PNG and JPEG Files

# Code Obfuscation

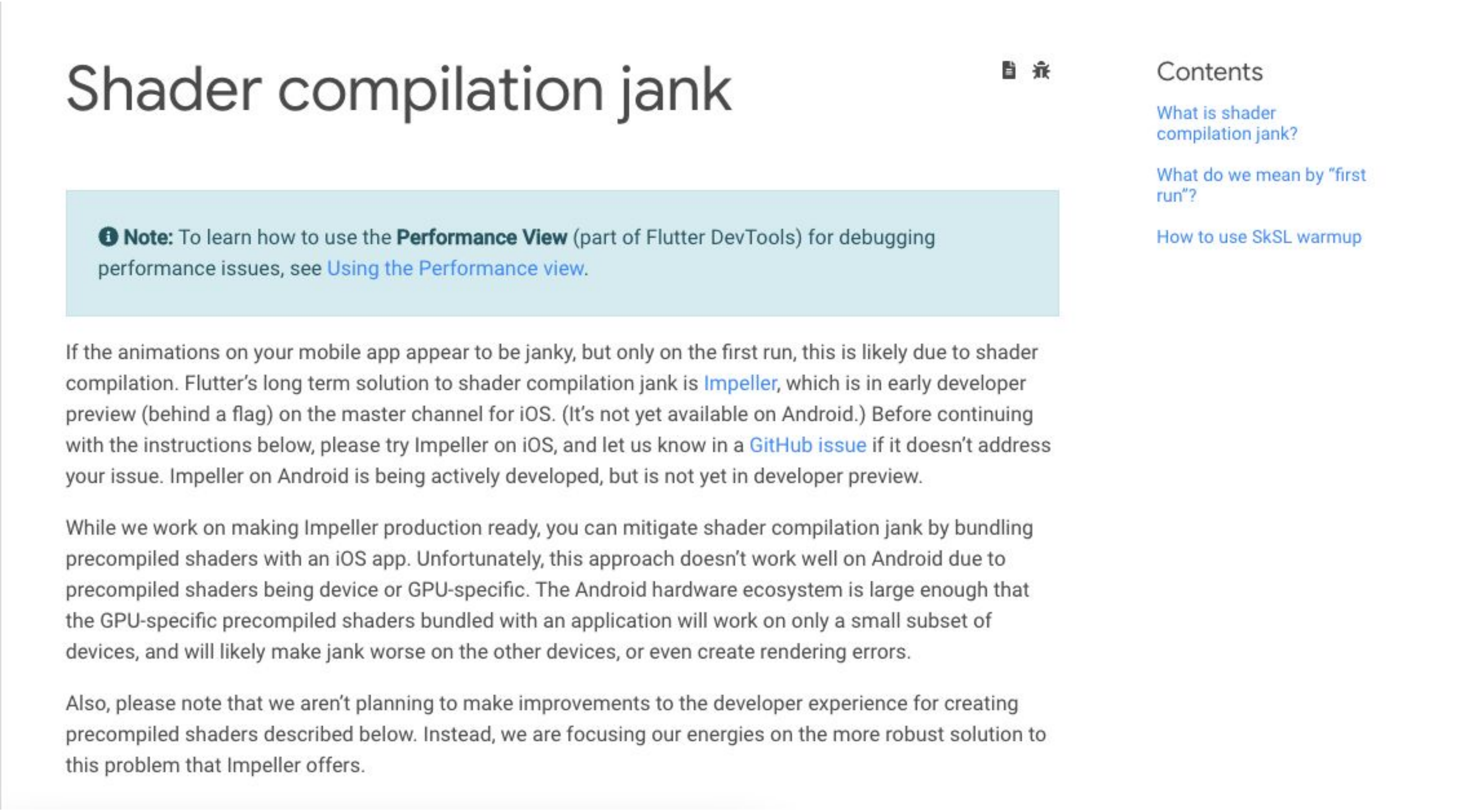
```
flutter build apk --obfuscate
--split-debug-info=/<project-name>/<directory>
```

Skooldio

# Demo

Skooldio

# Performance Optimization



# Shader compilation jank

> **Note:** To learn how to use the **Performance View** (part of Flutter DevTools) for debugging performance issues, see Using the Performance view.

If the animations on your mobile app appear to be janky, but only on the first run, this is likely due to shader compilation. Flutter's long term solution to shader compilation jank is Impeller, which is in early developer preview (behind a flag) on the master channel for iOS. (It's not yet available on Android.) Before continuing with the instructions below, please try Impeller on iOS, and let us know in a GitHub issue if it doesn't address your issue. Impeller on Android is being actively developed, but is not yet in developer preview.

While we work on making Impeller production ready, you can mitigate shader compilation jank by bundling precompiled shaders with an iOS app. Unfortunately, this approach doesn't work well on Android due to precompiled shaders being device or GPU-specific. The Android hardware ecosystem is large enough that the GPU-specific precompiled shaders bundled with an application will work on only a small subset of devices, and will likely make jank worse on the other devices, or even create rendering errors.

Also, please note that we aren't planning to make improvements to the developer experience for creating precompiled shaders described below. Instead, we are focusing our energies on the more robust solution to this problem that Impeller offers.

Contents

# Pre-Warmed Skia

```
$ flutter run --profile --cache-sksl

$ flutter build ios --bundle-sksl-path flutter_01.sksl.json
```

Skooldio

# Demo