Real world Flutter

Maksim Lin

MANICHORD

www.manichord.com

This is not a demo...

Topics

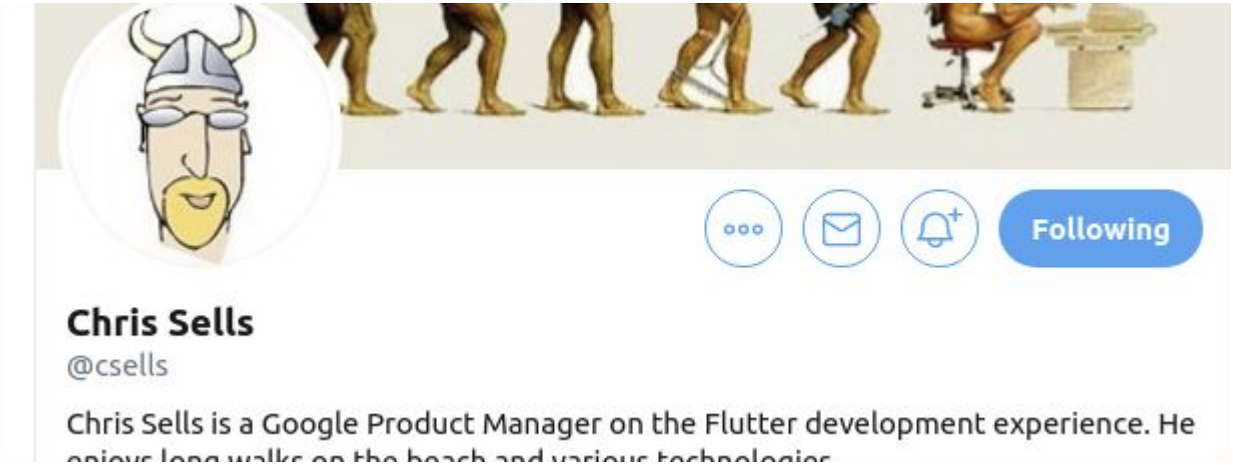
- Quick Intro
- Dev UX
- Porting
- State Management
- Architecture
- Project Structure
- Native Integration
- Logging / Debugging
- Monitoring Analytics

What is Flutter?

“Flutter is Google’s UI toolkit for building beautiful, natively compiled applications for ***mobile****,* ***web****, and* ***desktop*** *from a single codebase.”*

Developer UX

- First platform I have used where vendor seems to care *deeply* about:
  Developer UX
- a **PM** for developer ux:

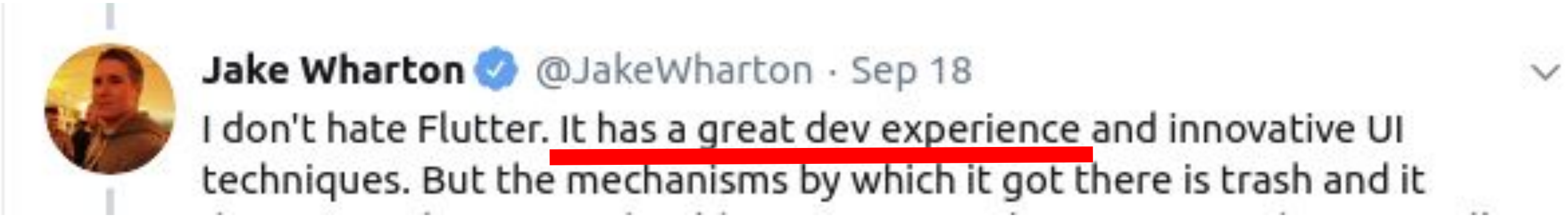


Chris Sells
@csells

Chris Sells is a Google Product Manager on the Flutter development experience. He enjoys long walks on the beach and various technologies.

Developer UX

Even the biggest critics agree...



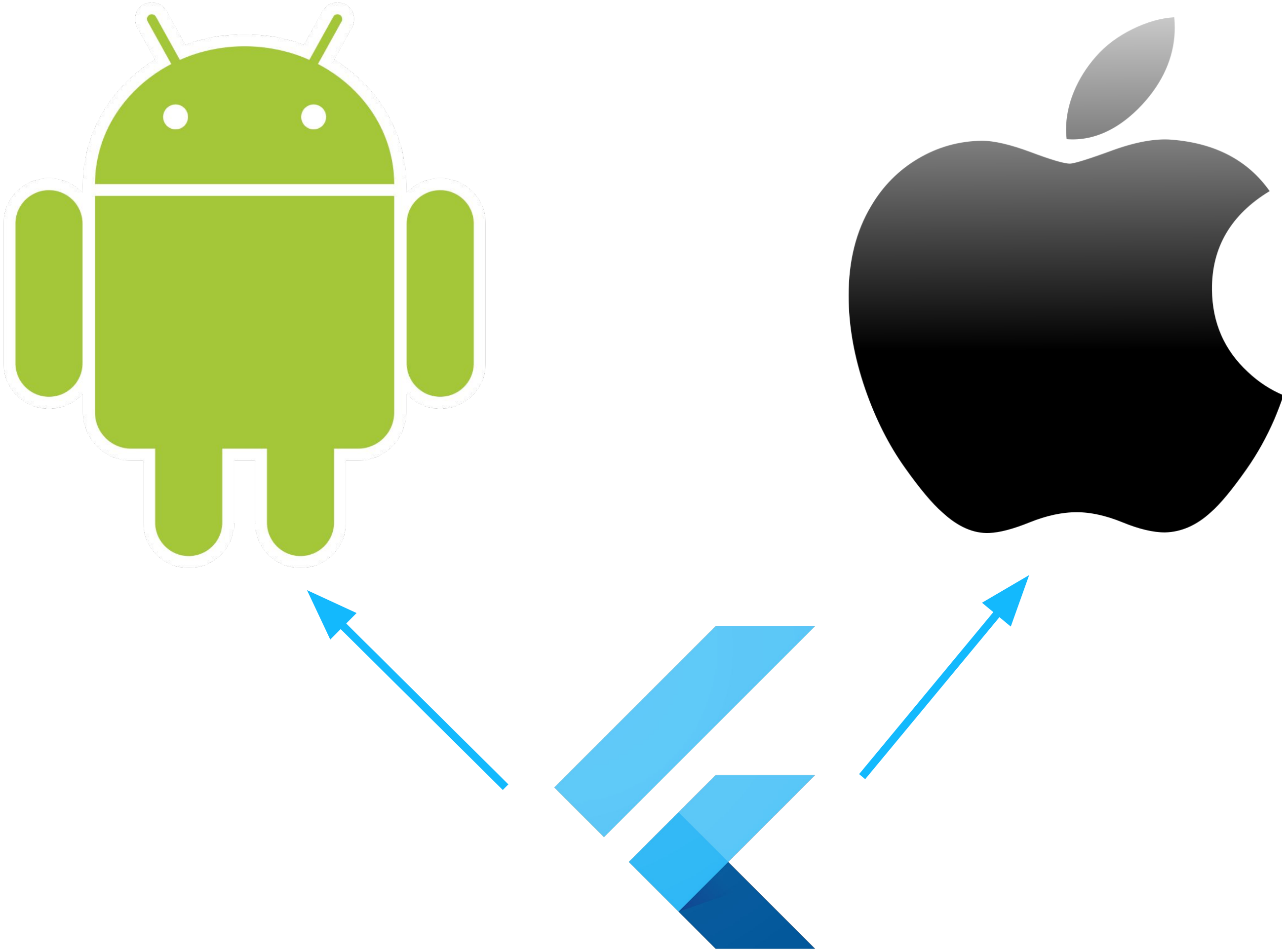
Jake Wharton @JakeWharton · Sep 18

I don't hate Flutter. It has a great dev experience and innovative UI techniques. But the mechanisms by which it got there is trash and it

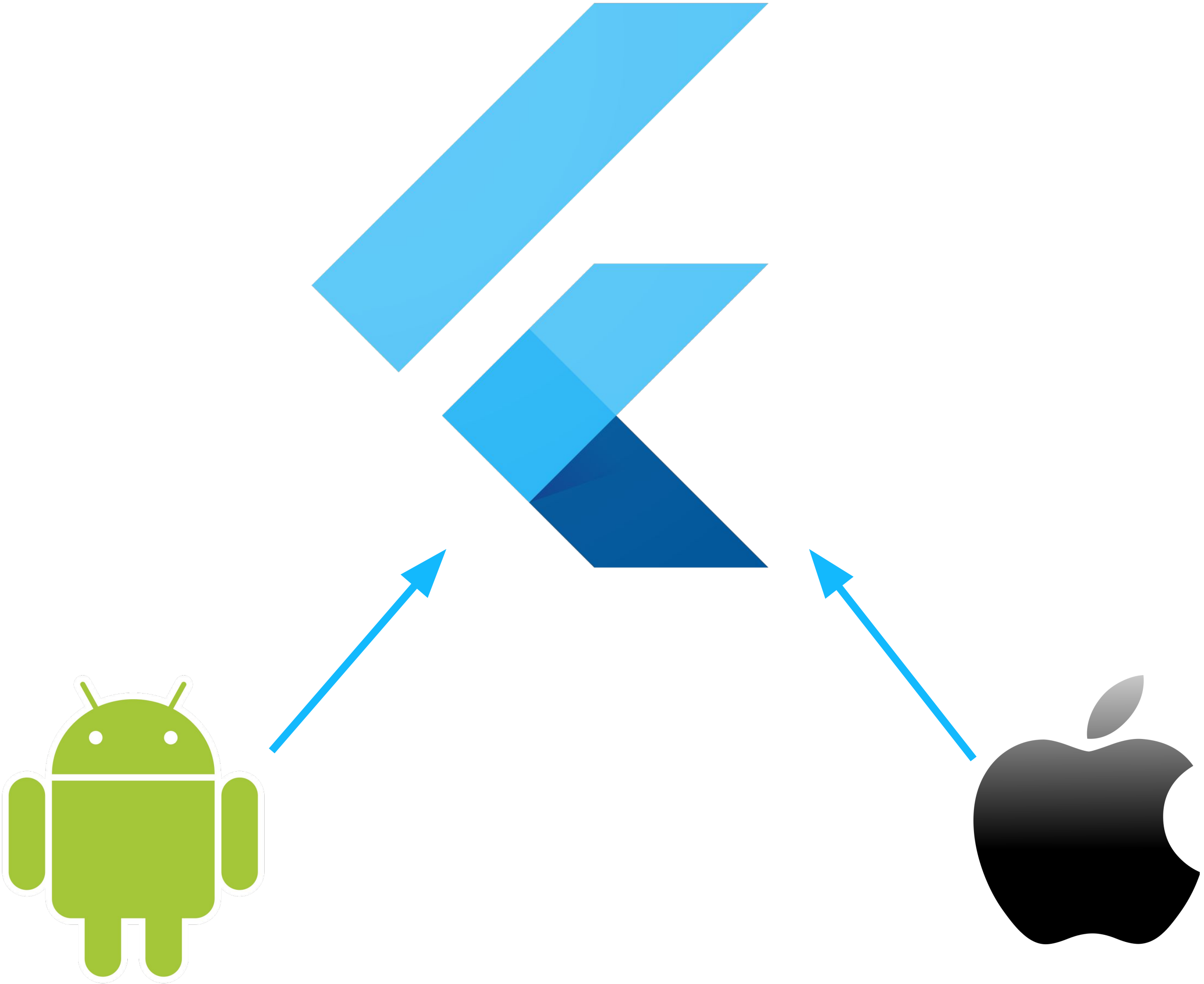
Flip, Flop, FLAP

Porting/Integrating existing code-bases

Google: *Add to App*

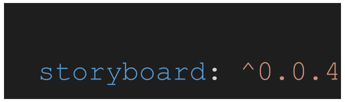


Maks: *FLAP*



Storyboarding

- Storyboard package:



```
storyboard: ^0.0.4
```

- “Container -> Content” widgets architecture
- Inspired by Storybook for web: React, Angular, etc

State management

Use Provider and BLOC’s
- Be happy and move on! 😉👌

- Keep all business logic out of widgets, put it in BLOCs (or anywhere else except widgets!)
- **Pragmatic** BLOC: everything **doesn't** have to be a stream!
- If doing “clean architecture” have BLOCs use services

State Mgmt: DECOUPLE from Blocs/Providers for ***generic*** widgets

Often can just pass in a stream and/or callbacks to actual widget from “container” widget to make easier to reuse and also to storyboard.

eg.

Yet another idea Flutter “borrowed” from React. Dan Abramov called it:

“Presentational and Container Components”

Project Structure

Just **my way**, unlike Android (***almost***) nothing is proscribed by Flutter

Top-level:

- lib/
- assets/
- test/
- docs/

Project Layout: lib/

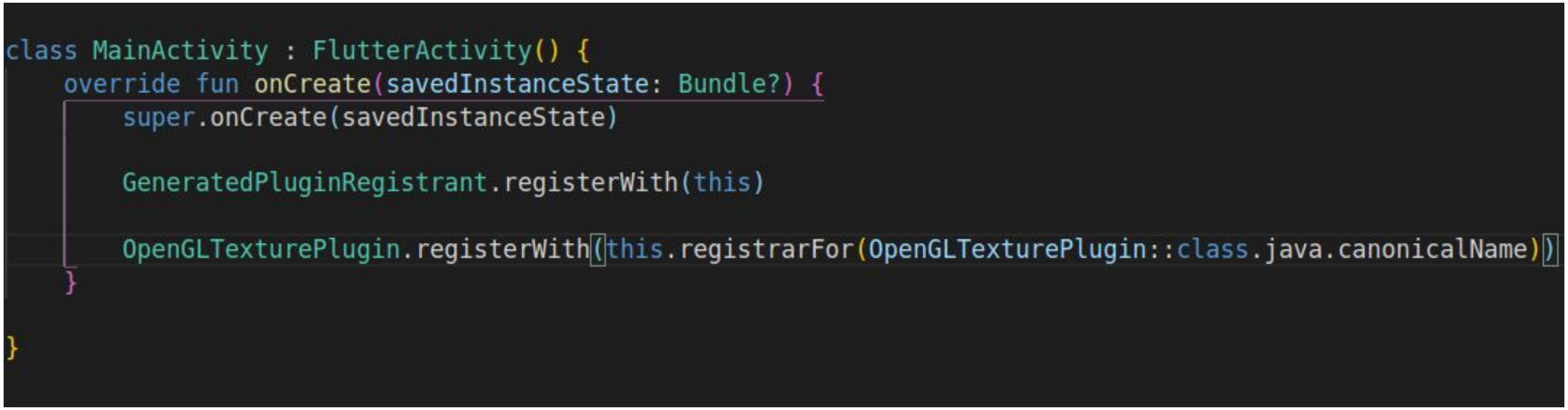
- blocs/
- debug/
- localisation/
- models/
- plugins/
- screens/
- services/
- style/
- types/
- widgets/

Project Layout: assets/

- data/
- fonts/
- l10n/
- images
  /icons
  /*{screen_name}*
- videos
  /*{screen_name}*

Native Integration

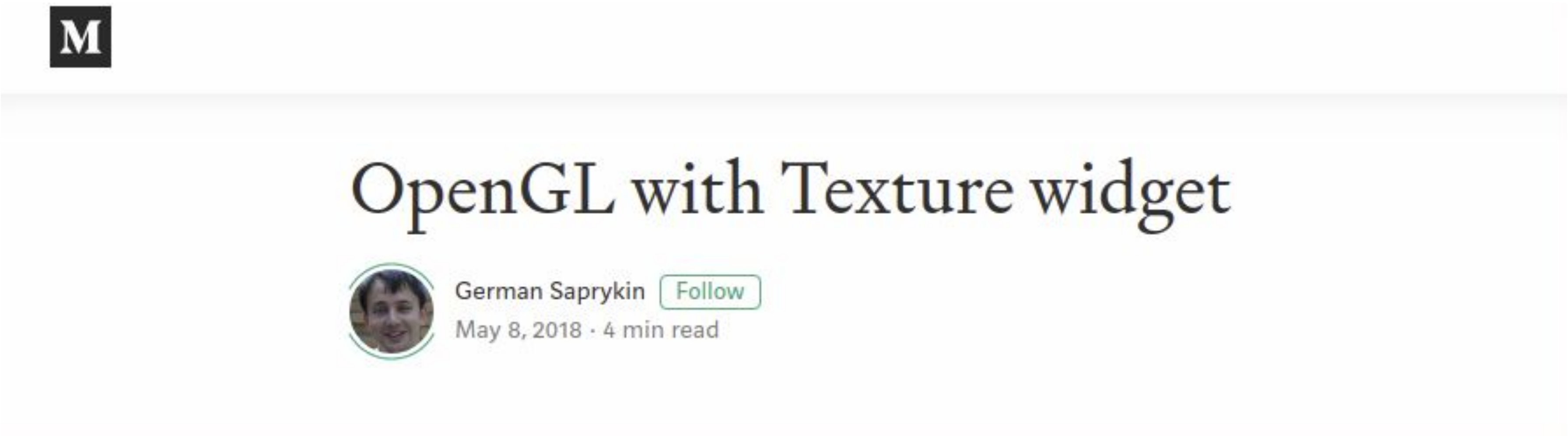
- Easiest to do directly without packaging custom plugins
- Need to manually initialise
  eg. for Android:



```
class MainActivity : FlutterActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        GeneratedPluginRegistrant.registerWith(this)

        OpenGLTexturePlugin.registerWith(this.registrarFor(OpenGLTexturePlugin::class.java.canonicalName))
    }

}
```

Deep Integration: OpenGL

- Flutter -> Java -> C++
- Flutter -> ObjC -> C++
- Oh my! Can it be done?
- Others paved the way…



OpenGL with Texture widget

German Saprykin

May 8, 2018 · 4 min read

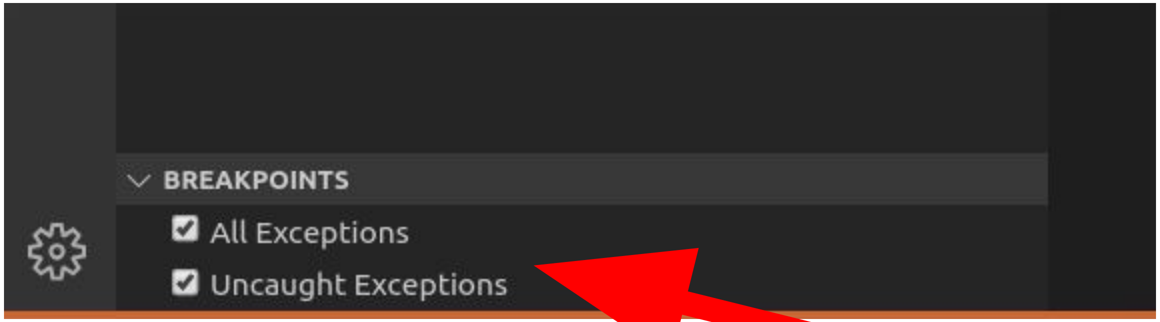
When things go wrong

We as developers spend most of our time with things that have gone wrong...

Logging

- Much more useful than with native dev as no recompile needed
- Lots to choose from!...
  stdout (print), package:logging,
  dart:developer, 3rd party pkgs
- Which to use?
- … print for quick/temporary,
- … dart:developer
- Cannot filter or set levels in VS Code or Devtools 😞

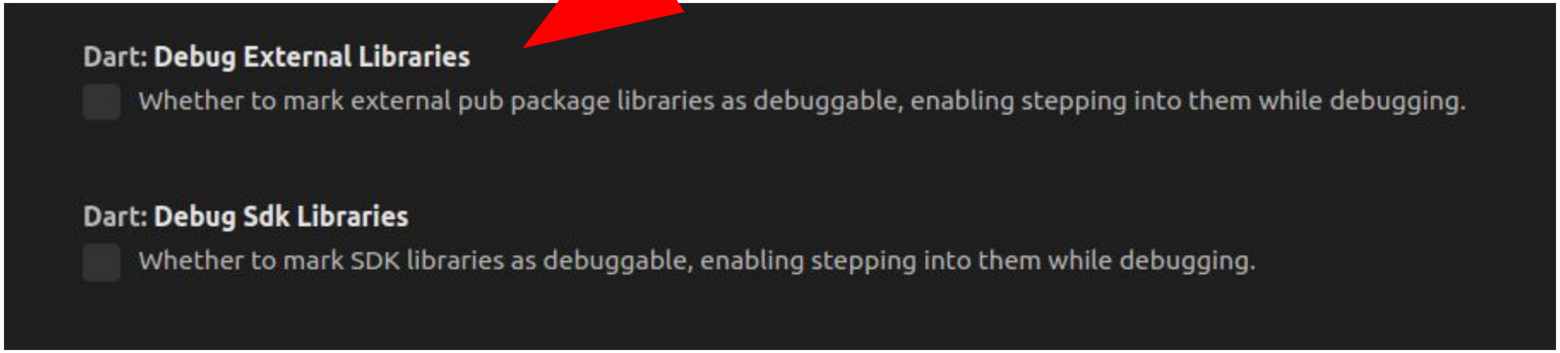
Debugging all the things...



BREAKPOINTS

☑ All Exceptions

☑ Uncaught Exceptions



Dart: **Debug External Libraries**

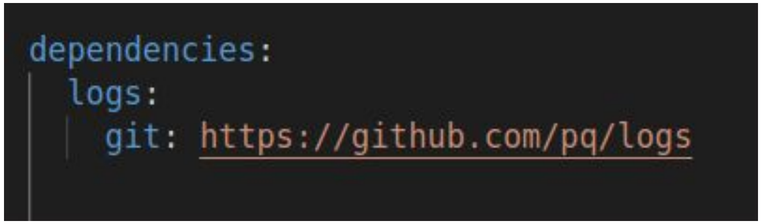
□ Whether to mark external pub package libraries as debuggable, enabling stepping into them while debugging.

Dart: **Debug Sdk Libraries**

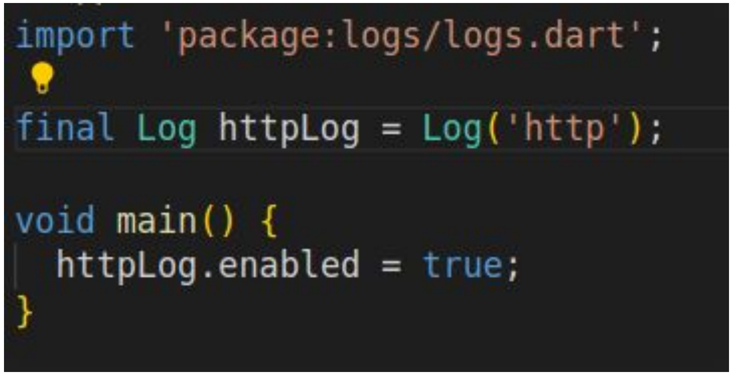
□ Whether to mark SDK libraries as debuggable, enabling stepping into them while debugging.

Bonus:
Network Logging & Debugging

Using Logging:

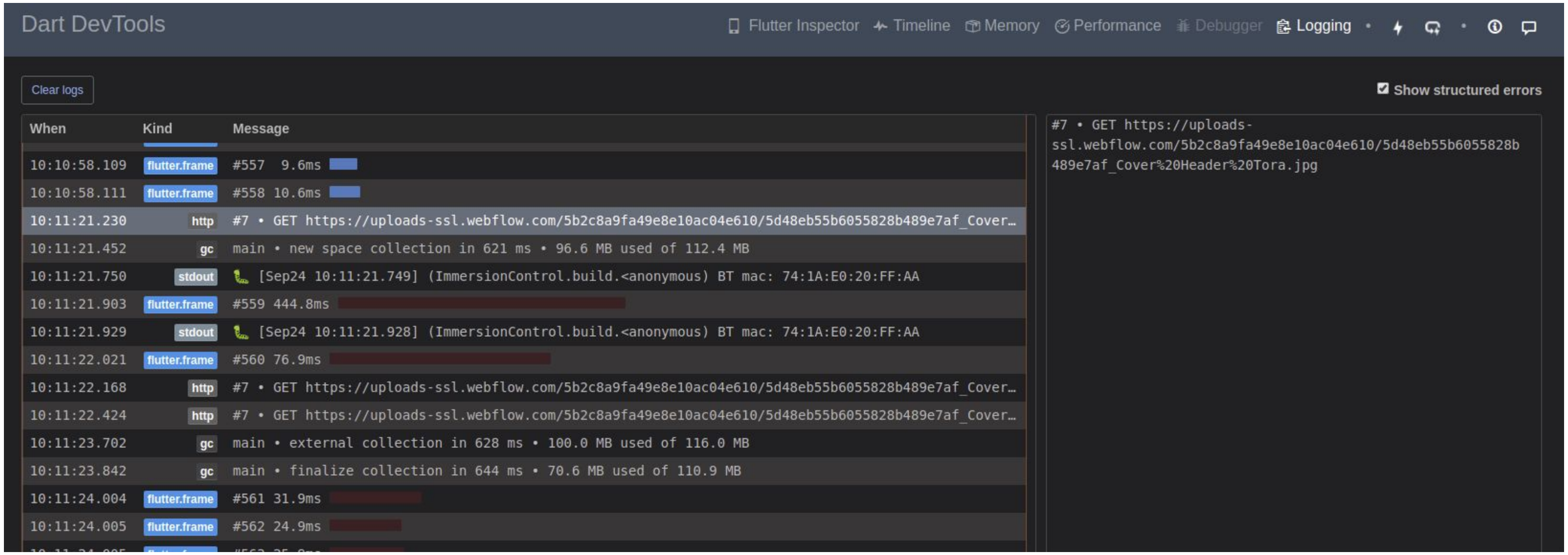


```
dependencies:
  logs:
    git: https://github.com/pq/logs
```



```
import 'package:logs/logs.dart';

final Log httpLog = Log('http');

void main() {
  httpLog.enabled = true;
}
```

Not really maintained now 😞

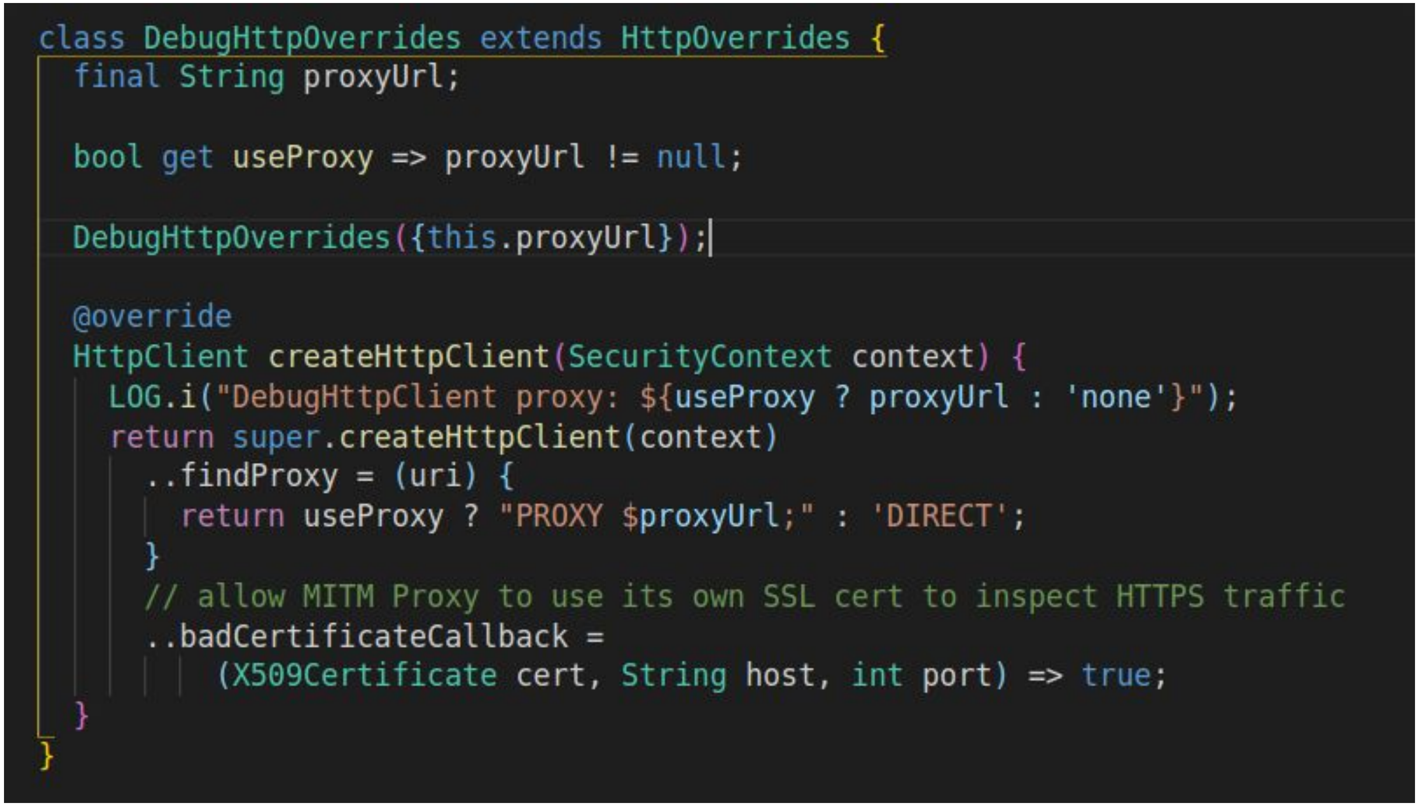


```
When          Kind           Message
10:10:58.109  flutter.frame  #557  9.6ms
10:10:58.111  flutter.frame  #558 10.6ms
10:11:21.230           http  #7 • GET https://uploads-ssl.webflow.com/5b2c8a9fa49e8e10ac04e610/5d48eb55b6055828b489e7af_Cover…
10:11:21.452             gc  main • new space collection in 621 ms • 96.6 MB used of 112.4 MB
10:11:21.750         stdout  [Sep24 10:11:21.749] (ImmersionControl.build.<anonymous>) BT mac: 74:1A:E0:20:FF:AA
10:11:21.903  flutter.frame  #559 444.8ms
10:11:21.929         stdout  [Sep24 10:11:21.928] (ImmersionControl.build.<anonymous>) BT mac: 74:1A:E0:20:FF:AA
10:11:22.021  flutter.frame  #560 76.9ms
10:11:22.168           http  #7 • GET https://uploads-ssl.webflow.com/5b2c8a9fa49e8e10ac04e610/5d48eb55b6055828b489e7af_Cover…
10:11:22.424           http  #7 • GET https://uploads-ssl.webflow.com/5b2c8a9fa49e8e10ac04e610/5d48eb55b6055828b489e7af_Cover…
10:11:23.702             gc  main • external collection in 628 ms • 100.0 MB used of 116.0 MB
10:11:23.842             gc  main • finalize collection in 644 ms • 70.6 MB used of 110.9 MB
10:11:24.004  flutter.frame  #561 31.9ms
10:11:24.005  flutter.frame  #562 24.9ms
```

```
#7 • GET https://uploads-
ssl.webflow.com/5b2c8a9fa49e8e10ac04e610/5d48eb55b6055828b
489e7af_Cover%20Header%20Tora.jpg
```

Bonus:
Network Logging & Debugging

Using Proxy (eg. Charles):



```
class DebugHttpOverrides extends HttpOverrides {
  final String proxyUrl;

  bool get useProxy => proxyUrl != null;

  DebugHttpOverrides({this.proxyUrl});

  @override
  HttpClient createHttpClient(SecurityContext context) {
    LOG.i("DebugHttpClient proxy: ${useProxy ? proxyUrl : 'none'}");
    return super.createHttpClient(context)
      ..findProxy = (uri) {
        return useProxy ? "PROXY $proxyUrl;" : 'DIRECT';
      }
      // allow MITM Proxy to use its own SSL cert to inspect HTTPS traffic
      ..badCertificateCallback =
          (X509Certificate cert, String host, int port) => true;
  }
}
```

To Fork or not to Fork...

“Fork in the Road” by Curtis Gregory Perry - CC BY-NC-SA 2.0

Even first-party plugins can need more functionality (markdown, sentry)

Crashlytics: Dont!

- Crashlytics just *painful*, poor functionality
- Use Sentry instead!
- Flutter docs recommends it over Crashlytics!
- First-party Sentry package!
- still under construction - PR incoming for context support...

Monitoring: Errors and Analytics

Thank You!

Questions?

@mklin

maks