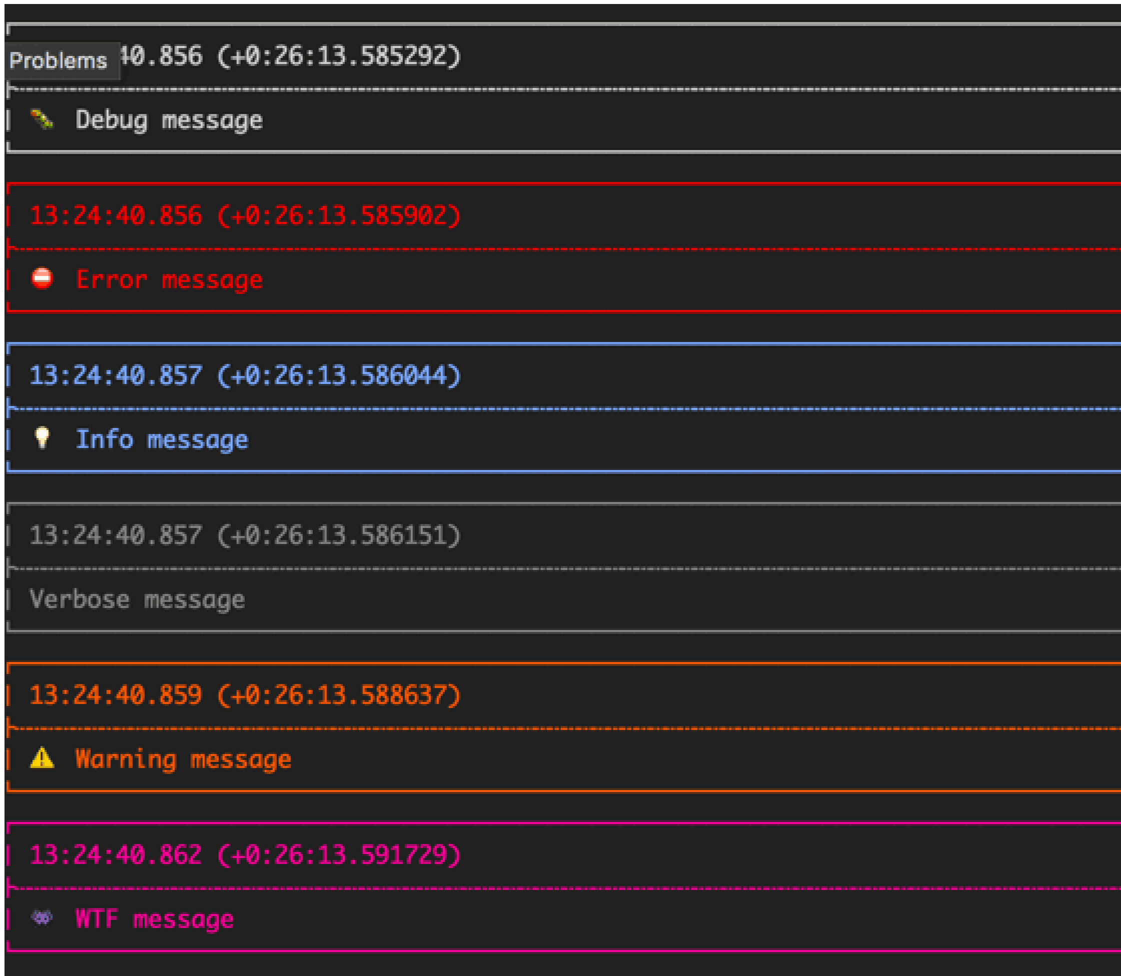
Logging



```
Problems 40.856 (+0:26:13.585292)
Debug message

13:24:40.856 (+0:26:13.585902)
Error message

13:24:40.857 (+0:26:13.586044)
Info message

13:24:40.857 (+0:26:13.586151)
Verbose message

13:24:40.859 (+0:26:13.588637)
Warning message

13:24:40.862 (+0:26:13.591729)
WTF message
```

Using the Logging view

Tools > DevTools > Using the Logging view

Note: The logging view works with all Flutter and Dart applications.

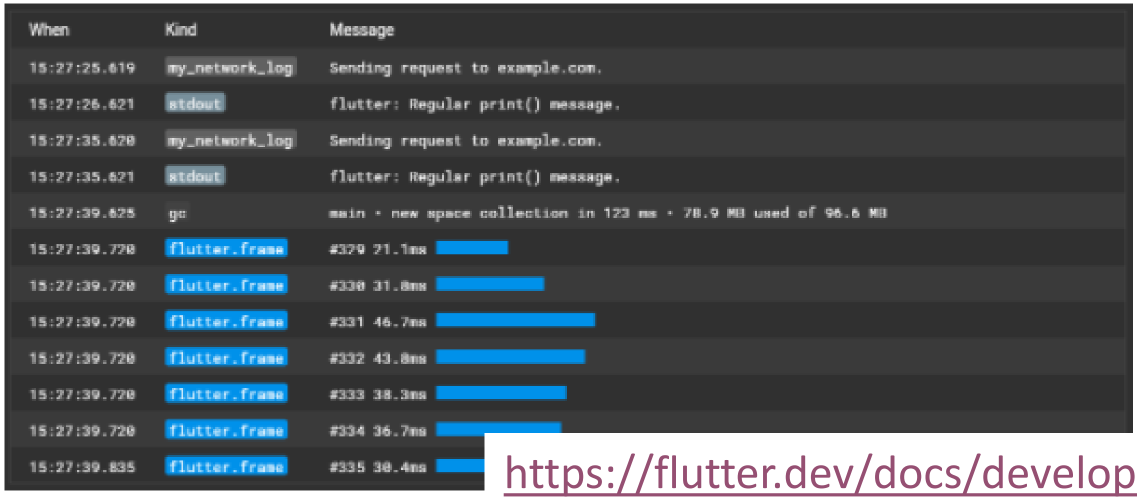
What is it?

The logging view displays events from the Dart runtime, application frameworks (like Flutter), and application-level logging events.

Standard logging events

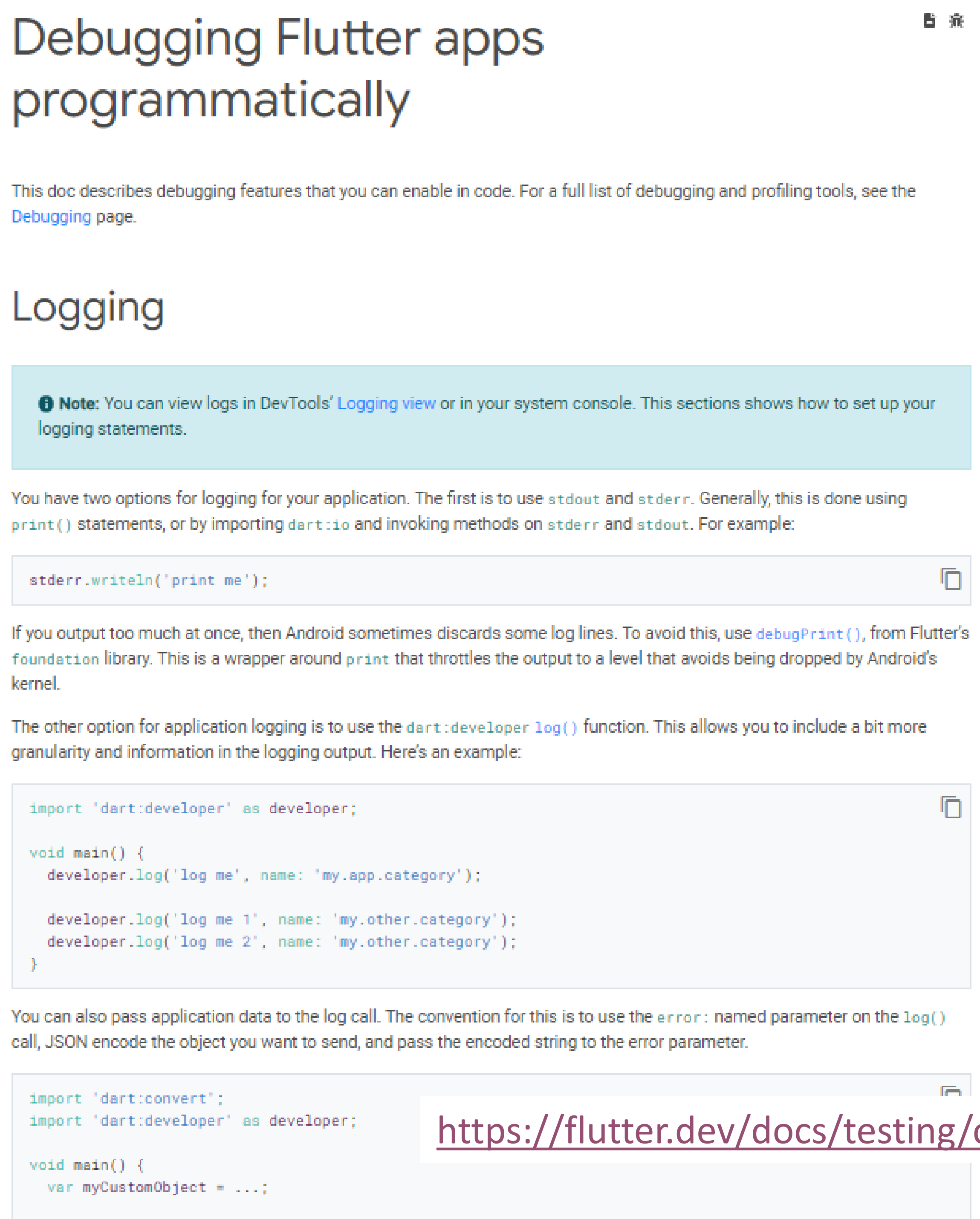
By default, the logging view shows:

- Garbage collection events from the Dart runtime
- Flutter framework events, like frame creation events
- `stdout` and `stderr` from applications
- Custom logging events from applications



```
When          Kind              Message
15:27:25.619  my_network_log    Sending request to example.com.
15:27:26.621  stdout            flutter: Regular print() message.
15:27:35.620  my_network_log    Sending request to example.com.
15:27:35.621  stdout            flutter: Regular print() message.
15:27:39.625  gc                main • new space collection in 123 ms • 78.9 MB used of 96.6 MB
15:27:39.720  flutter.frame     #329 21.1ms
15:27:39.720  flutter.frame     #330 31.8ms
15:27:39.720  flutter.frame     #331 46.7ms
15:27:39.720  flutter.frame     #332 43.8ms
15:27:39.720  flutter.frame     #333 38.3ms
15:27:39.720  flutter.frame     #334 36.7ms
15:27:39.835  flutter.frame     #335 30.4ms
```

https://flutter.dev/docs/development/tools/devtools/logging



Debugging Flutter apps programmatically

This doc describes debugging features that you can enable in code. For a full list of debugging and profiling tools, see the Debugging page.

Logging

Note: You can view logs in DevTools' Logging view or in your system console. This sections shows how to set up your logging statements.

You have two options for logging for your application. The first is to use `stdout` and `stderr`. Generally, this is done using `print()` statements, or by importing `dart:io` and invoking methods on `stderr` and `stdout`. For example:

```
stderr.writeln('print me');
```

If you output too much at once, then Android sometimes discards some log lines. To avoid this, use `debugPrint()`, from Flutter's `foundation` library. This is a wrapper around `print` that throttles the output to a level that avoids being dropped by Android's kernel.

The other option for application logging is to use the `dart:developer` `log()` function. This allows you to include a bit more granularity and information in the logging output. Here's an example:

```
import 'dart:developer' as developer;

void main() {
  developer.log('log me', name: 'my.app.category');

  developer.log('log me 1', name: 'my.other.category');
  developer.log('log me 2', name: 'my.other.category');
}
```

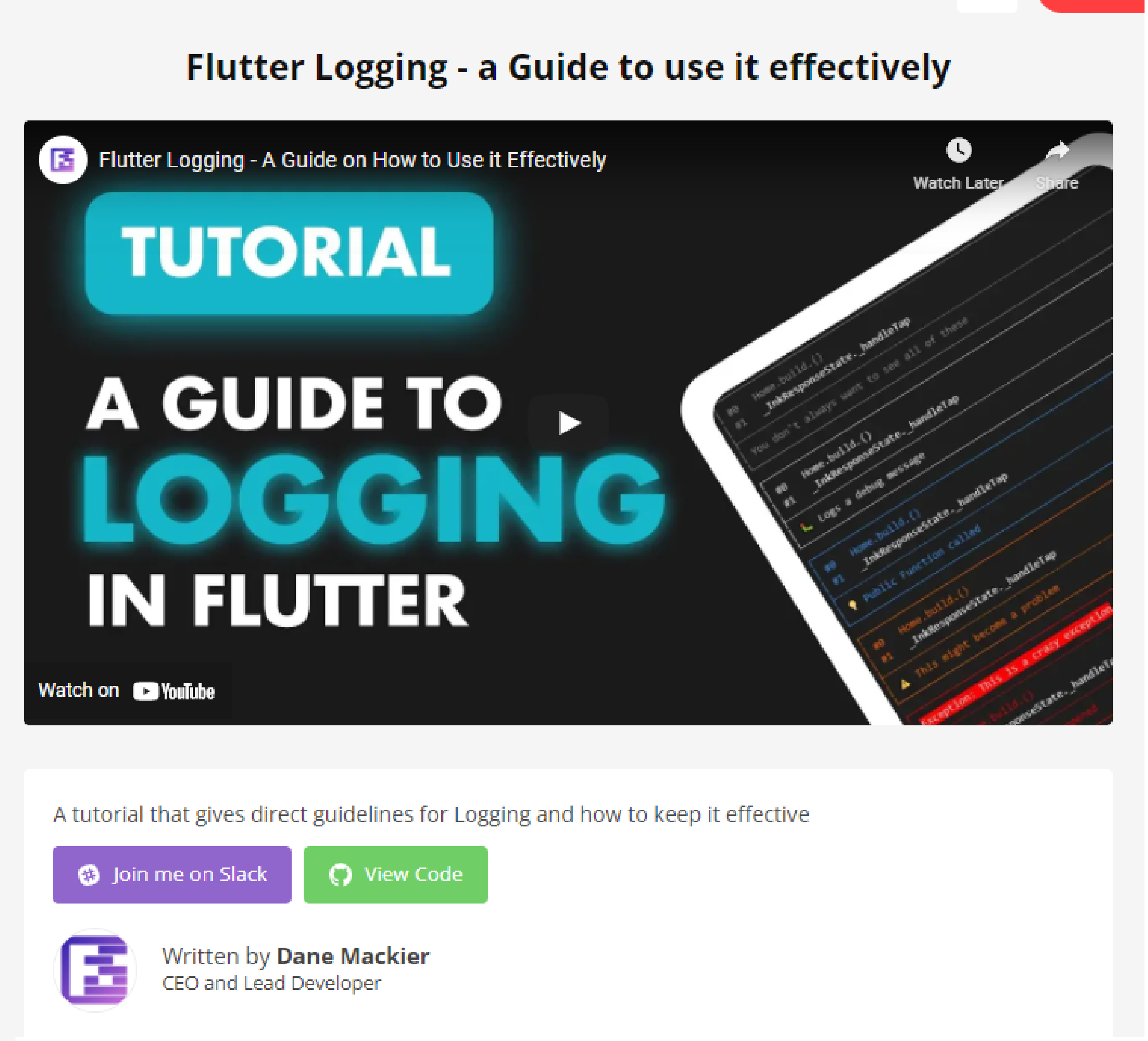
You can also pass application data to the log call. The convention for this is to use the `error:` named parameter on the `log()` call, JSON encode the object you want to send, and pass the encoded string to the error parameter.

```
import 'dart:convert';
import 'dart:developer' as developer;

void main() {
  var myCustomObject = ...;
```

Flutter

https://flutter.dev/docs/testing/code-debugging

universidade de aveiro



Flutter Logging - a Guide to use it effectively

```
#0 Home.build.()
#1 _InkResponseState._handleTap
You don't always want to see all of these
#0 Home.build.()
#1 _InkResponseState._handleTap
Logs a debug message
#0 Home.build.()
#1 _InkResponseState._handleTap
Public Function called
#0 Home.build.()
#1 _InkResponseState._handleTap
This might become a problem
Exception: This is a crazy exception
```

A tutorial that gives direct guidelines for Logging and how to keep it effective

Written by **Dane Mackier**
CEO and Lead Developer

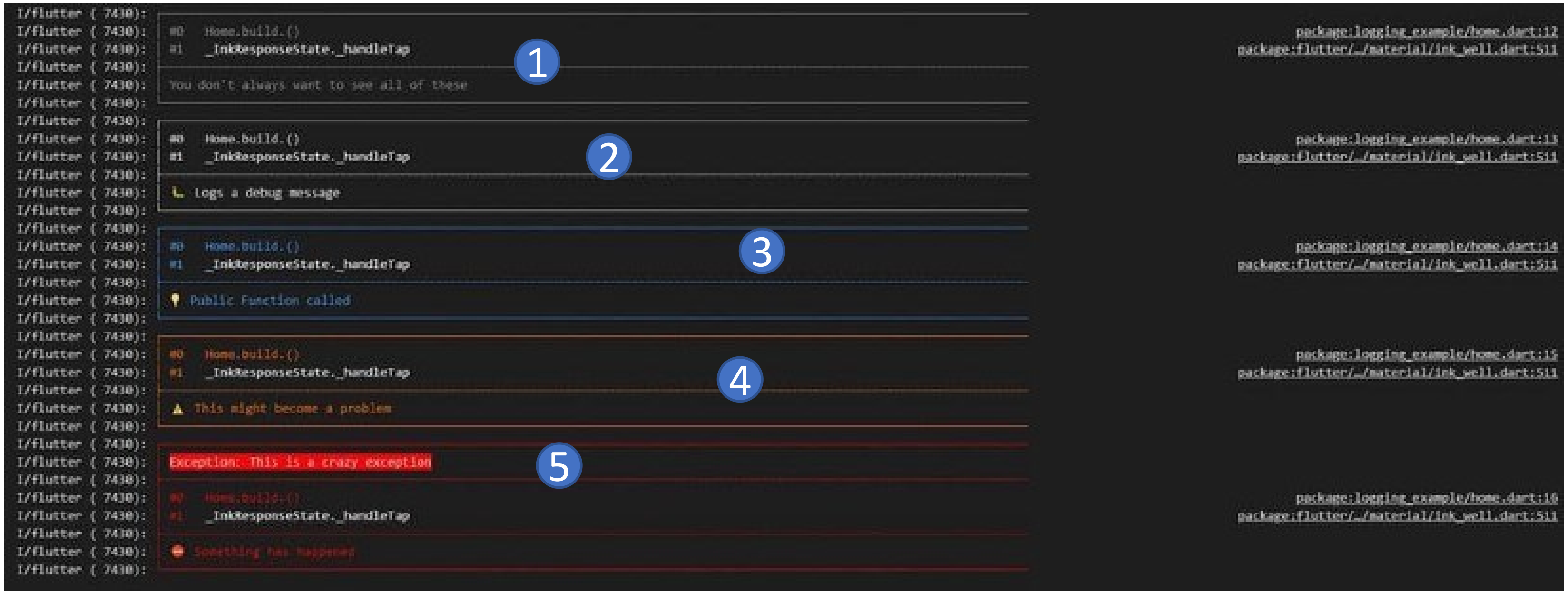
https://www.filledstacks.com/post/flutter-logging-a-guide-to-use-it-effectively/

A Guide to Setting up Better Logging in Flutter

Dane Mackier Follow
Jul 2, 2019 · 4 min read

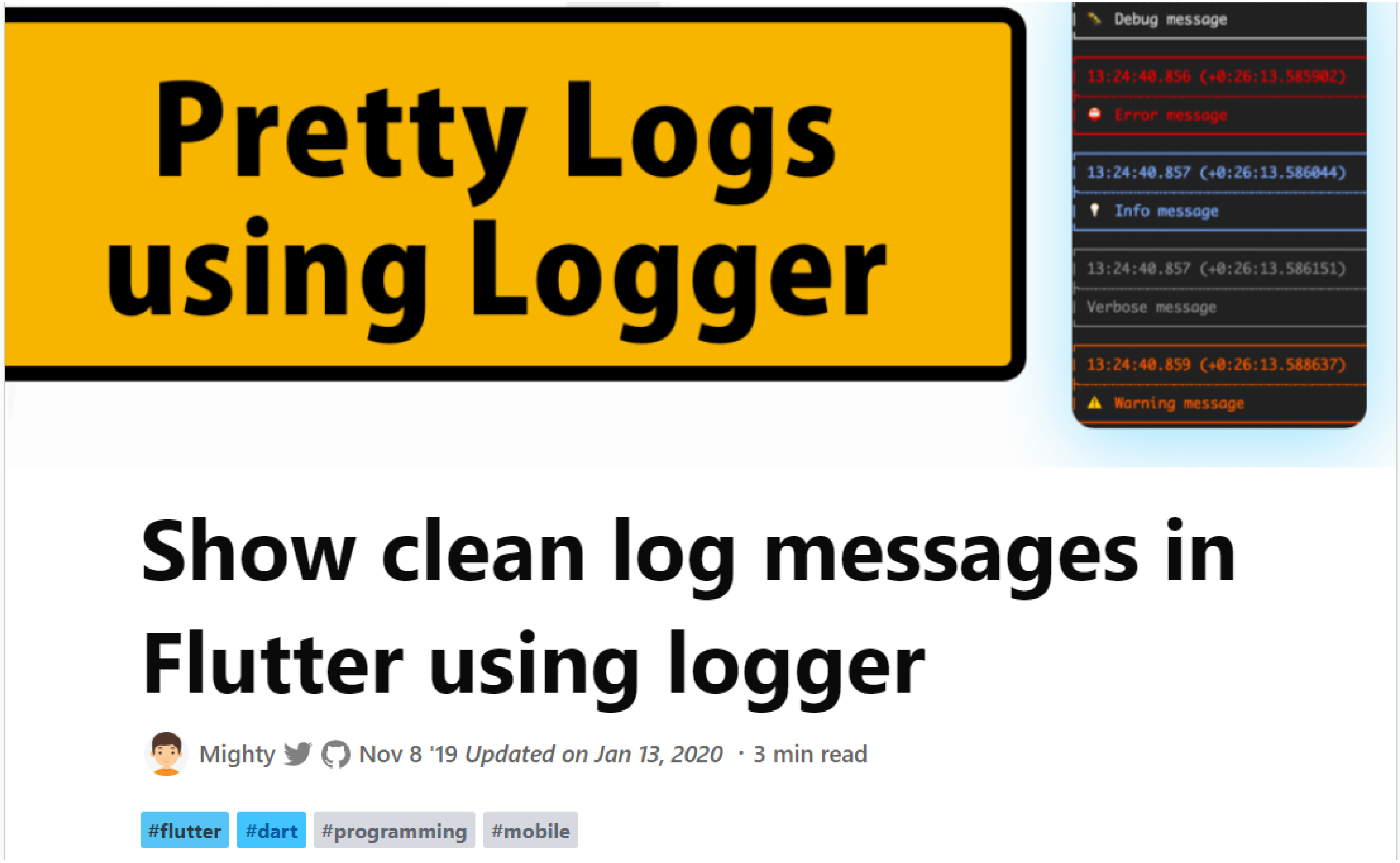
```
final logger = Logger();

logger.v('You don\'t always want to see all of these');   1
2 logger.d('Logs a debug message');
logger.i('Public Function called');   3
4 logger.w('This might become a problem');
logger.e('Something has happened');   5
```

Using a logger in code the entries appear in the flutter app logging view



```
I/flutter ( 7430):
I/flutter ( 7430):  #0   Home.build.()                                     package:logging_example/home.dart:12
I/flutter ( 7430):  #1   _InkResponseState._handleTap               package:flutter/…/material/ink_well.dart:511
I/flutter ( 7430):
I/flutter ( 7430):  You don't always want to see all of these
I/flutter ( 7430):
I/flutter ( 7430):
I/flutter ( 7430):  #0   Home.build.()                                     package:logging_example/home.dart:13
I/flutter ( 7430):  #1   _InkResponseState._handleTap               package:flutter/…/material/ink_well.dart:511
I/flutter ( 7430):
I/flutter ( 7430):  Logs a debug message
I/flutter ( 7430):
I/flutter ( 7430):
I/flutter ( 7430):  #0   Home.build.()                                     package:logging_example/home.dart:14
I/flutter ( 7430):  #1   _InkResponseState._handleTap               package:flutter/…/material/ink_well.dart:511
I/flutter ( 7430):
I/flutter ( 7430):  Public Function called
I/flutter ( 7430):
I/flutter ( 7430):
I/flutter ( 7430):  #0   Home.build.()                                     package:logging_example/home.dart:15
I/flutter ( 7430):  #1   _InkResponseState._handleTap               package:flutter/…/material/ink_well.dart:511
I/flutter ( 7430):
I/flutter ( 7430):  ⚠ This might become a problem
I/flutter ( 7430):
I/flutter ( 7430):
I/flutter ( 7430):  Exception: This is a crazy exception
I/flutter ( 7430):
I/flutter ( 7430):  #0   Home.build.()                                     package:logging_example/home.dart:16
I/flutter ( 7430):  #1   _InkResponseState._handleTap               package:flutter/…/material/ink_well.dart:511
I/flutter ( 7430):
I/flutter ( 7430):  ⛔ Something has happened
I/flutter ( 7430):
```

https://medium.com/flutter-community/a-guide-to-setting-up-better-logging-in-flutter-3db8bab2000e



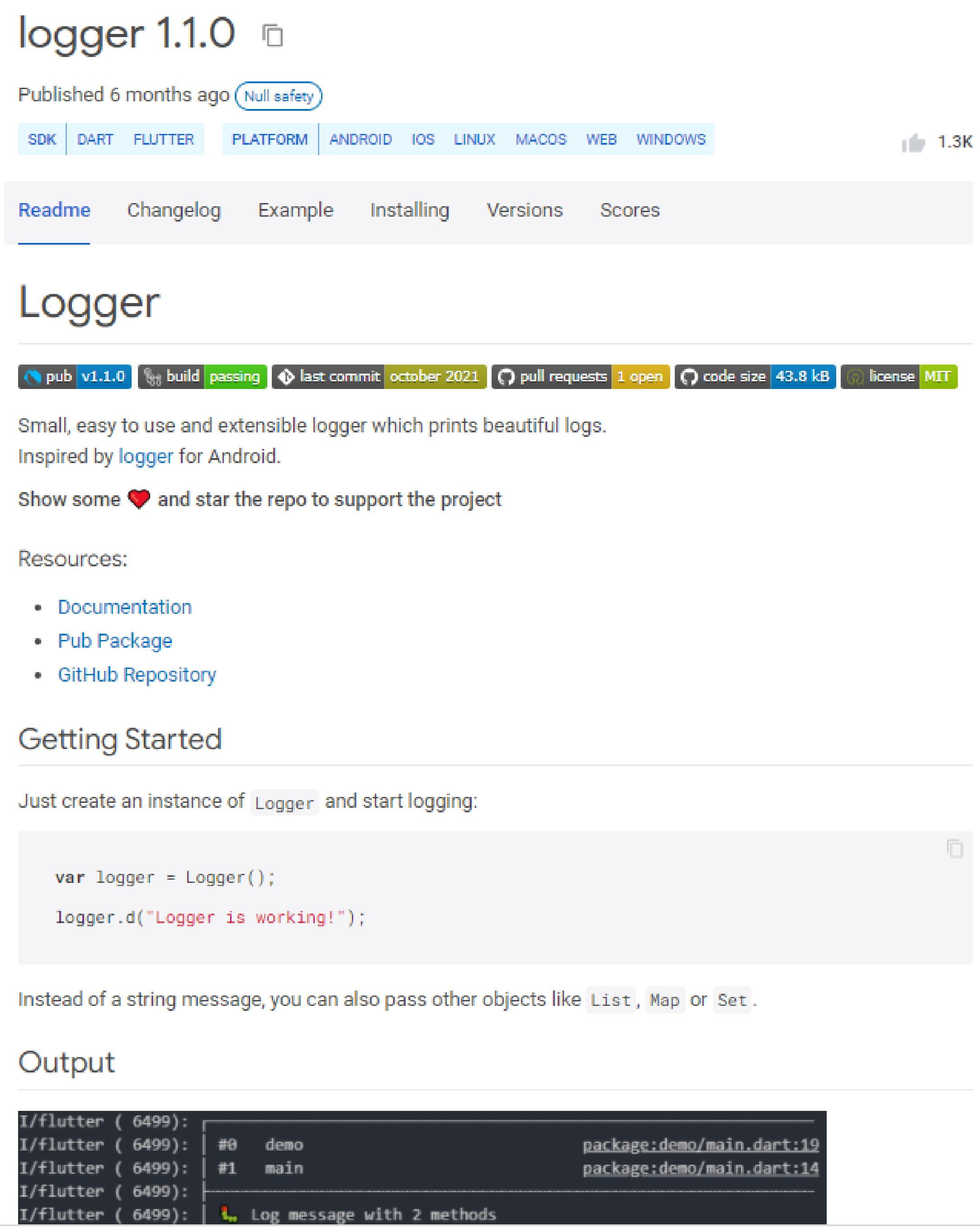
```
🐛 Debug message

13:24:40.856 (+0:26:13.585902)
⛔ Error message

13:24:40.857 (+0:26:13.586044)
💡 Info message

13:24:40.857 (+0:26:13.586151)
Verbose message

13:24:40.859 (+0:26:13.588637)
⚠ Warning message
```

Show clean log messages in Flutter using logger

Mighty Nov 8 '19 *Updated on Jan 13, 2020* · 3 min read

#flutter #dart #programming #mobile

https://dev.to/mightytechno/show-clean-log-messages-in-flutter-using-logger-4n23



logger 1.1.0

Published 6 months ago Null safety

SDK | DART FLUTTER   PLATFORM | ANDROID IOS LINUX MACOS WEB WINDOWS   1.3K

Readme   Changelog   Example   Installing   Versions   Scores

Logger

pub v1.1.0 | build passing | last commit october 2021 | pull requests 1 open | code size 43.8 kB | license MIT

Small, easy to use and extensible logger which prints beautiful logs.
Inspired by logger for Android.

Show some ❤️ and star the repo to support the project

Resources:

- Documentation
- Pub Package
- GitHub Repository

Getting Started

Just create an instance of `Logger` and start logging:

```
var logger = Logger();

logger.d("Logger is working!");
```

Instead of a string message, you can also pass other objects like `List`, `Map` or `Set`.

Output

```
I/flutter ( 6499): ┌──────────────────────────────────────────────────────────
I/flutter ( 6499): │ #0   demo                   package:demo/main.dart:19
I/flutter ( 6499): │ #1   main                   package:demo/main.dart:14
I/flutter ( 6499): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
I/flutter ( 6499): │ 🐛 Log message with 2 methods
```

logger 0.8.3

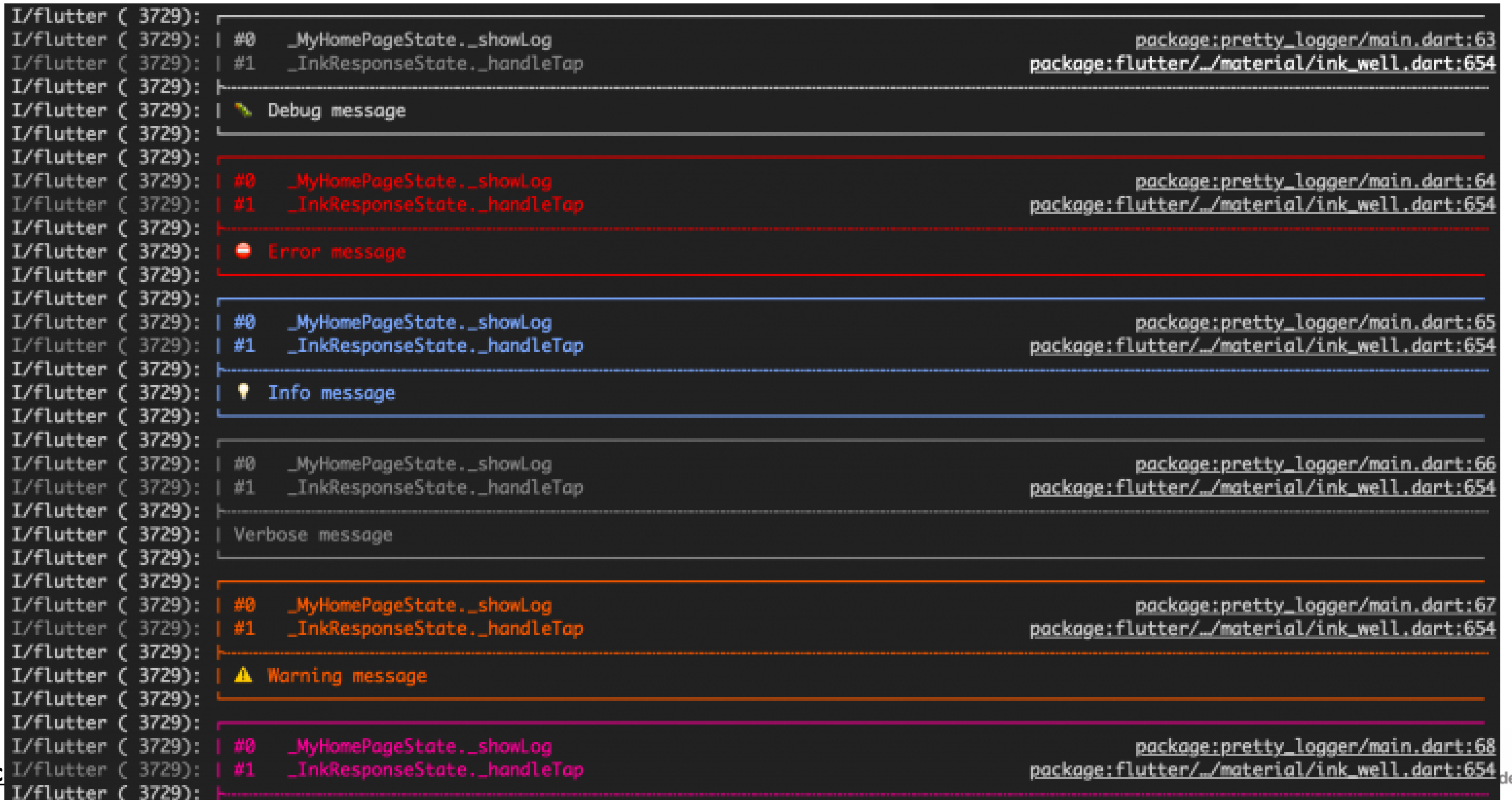
Published Jan 14, 2020 57 likes

DART NATIVE FLUTTER ANDROID IOS

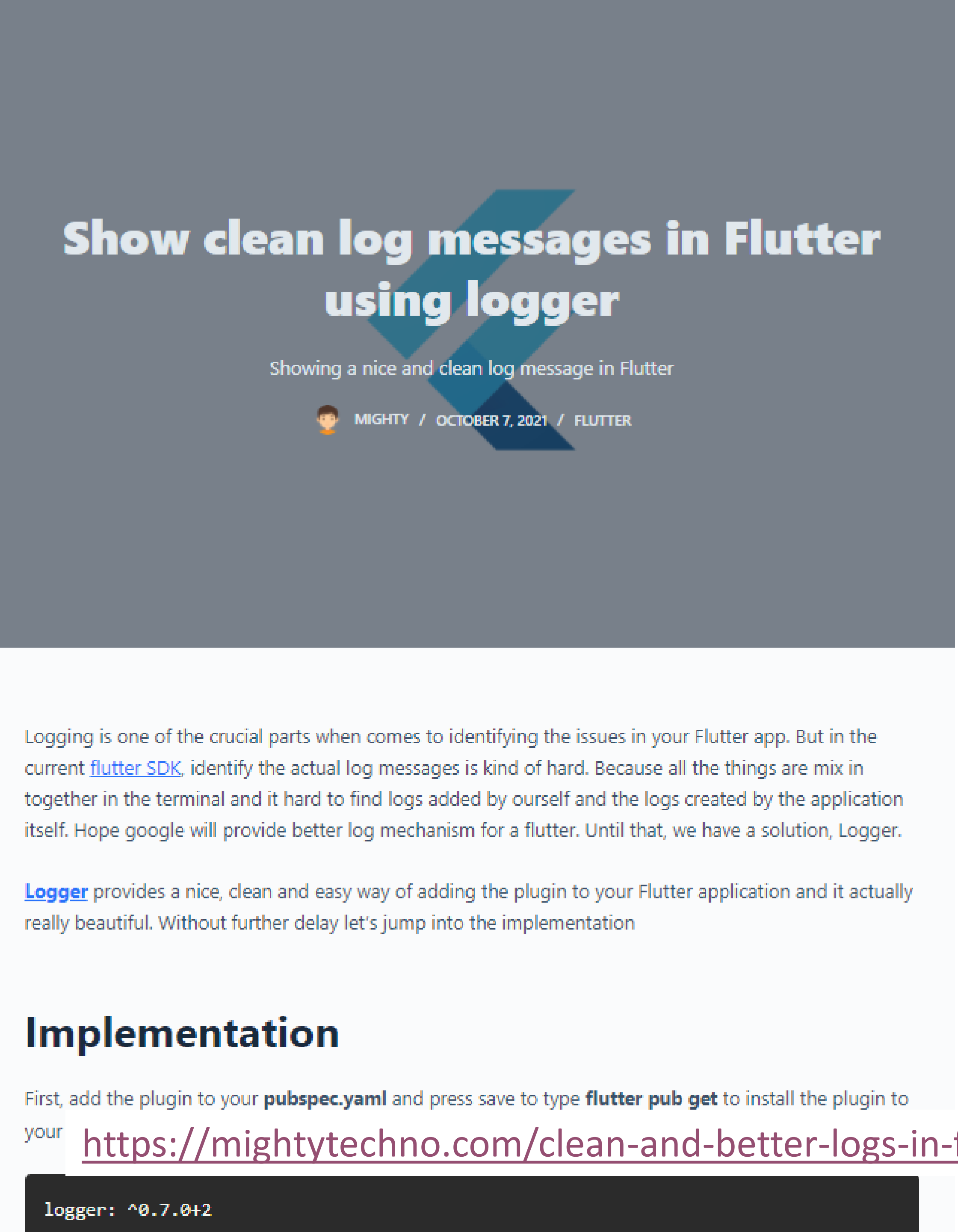
First, add the plugin to your **pubspec.yaml**

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

```dart
var logger = Logger();
logger.d("Debug message");
logger.e("Error message");
logger.i("Info message");
logger.v("Verbose message");
logger.w("Warning message");
logger.wtf("WTF message");
```



```
I/flutter ( 3729): ┌──────────────────────────────────────────────────────────────────
I/flutter ( 3729): │ #0   _MyHomePageState._showLog                  package:pretty_logger/main.dart:63
I/flutter ( 3729): │ #1   _InkResponseState._handleTap     package:flutter/…/material/ink_well.dart:654
I/flutter ( 3729): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
I/flutter ( 3729): │ 🐛 Debug message
I/flutter ( 3729): └──────────────────────────────────────────────────────────────────
I/flutter ( 3729): ┌──────────────────────────────────────────────────────────────────
I/flutter ( 3729): │ #0   _MyHomePageState._showLog                  package:pretty_logger/main.dart:64
I/flutter ( 3729): │ #1   _InkResponseState._handleTap     package:flutter/…/material/ink_well.dart:654
I/flutter ( 3729): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
I/flutter ( 3729): │ ⛔ Error message
I/flutter ( 3729): └──────────────────────────────────────────────────────────────────
I/flutter ( 3729): ┌──────────────────────────────────────────────────────────────────
I/flutter ( 3729): │ #0   _MyHomePageState._showLog                  package:pretty_logger/main.dart:65
I/flutter ( 3729): │ #1   _InkResponseState._handleTap     package:flutter/…/material/ink_well.dart:654
I/flutter ( 3729): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
I/flutter ( 3729): │ 💡 Info message
I/flutter ( 3729): └──────────────────────────────────────────────────────────────────
I/flutter ( 3729): ┌──────────────────────────────────────────────────────────────────
I/flutter ( 3729): │ #0   _MyHomePageState._showLog                  package:pretty_logger/main.dart:66
I/flutter ( 3729): │ #1   _InkResponseState._handleTap     package:flutter/…/material/ink_well.dart:654
I/flutter ( 3729): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
I/flutter ( 3729): │ Verbose message
I/flutter ( 3729): └──────────────────────────────────────────────────────────────────
I/flutter ( 3729): ┌──────────────────────────────────────────────────────────────────
I/flutter ( 3729): │ #0   _MyHomePageState._showLog                  package:pretty_logger/main.dart:67
I/flutter ( 3729): │ #1   _InkResponseState._handleTap     package:flutter/…/material/ink_well.dart:654
I/flutter ( 3729): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
I/flutter ( 3729): │ ⚠ Warning message
I/flutter ( 3729): └──────────────────────────────────────────────────────────────────
I/flutter ( 3729): ┌──────────────────────────────────────────────────────────────────
I/flutter ( 3729): │ #0   _MyHomePageState._showLog                  package:pretty_logger/main.dart:68
I/flutter ( 3729): │ #1   _InkResponseState._handleTap     package:flutter/…/material/ink_well.dart:654
I/flutter ( 3729): ├┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄
```



Show clean log messages in Flutter using logger

Showing a nice and clean log message in Flutter

MIGHTY / OCTOBER 7, 2021 / FLUTTER

Logging is one of the crucial parts when comes to identifying the issues in your Flutter app. But in the current flutter SDK, identify the actual log messages is kind of hard. Because all the things are mix in together in the terminal and it hard to find logs added by ourself and the logs created by the application itself. Hope google will provide better log mechanism for a flutter. Until that, we have a solution, Logger.

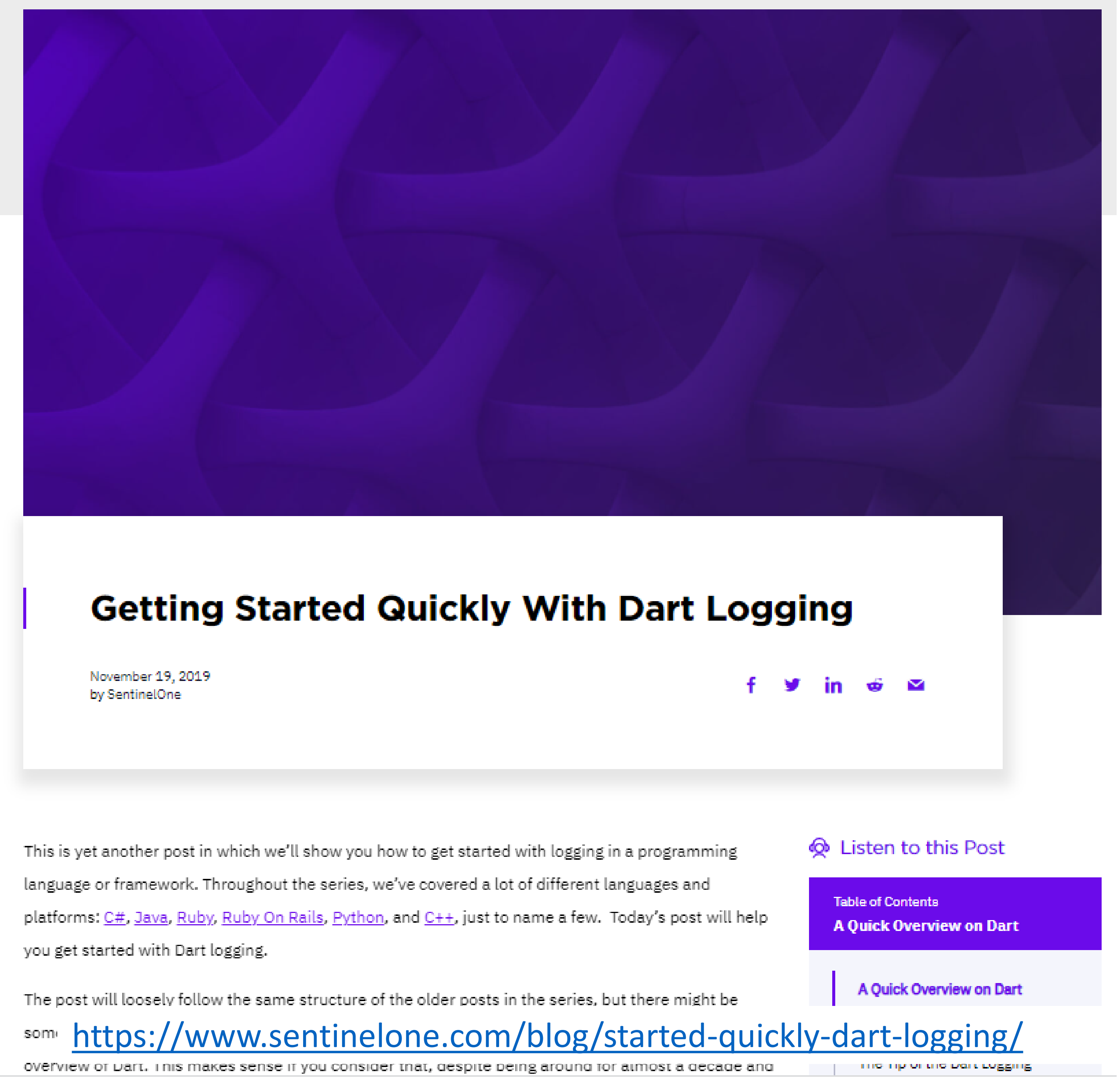
Logger provides a nice, clean and easy way of adding the plugin to your Flutter application and it actually really beautiful. Without further delay let's jump into the implementation

Implementation

First, add the plugin to your **pubspec.yaml** and press save to type **flutter pub get** to install the plugin to your

```
logger: ^0.7.0+2
```

https://mightytechno.com/clean-and-better-logs-in-flutter-using-logger/



Getting Started Quickly With Dart Logging

November 19, 2019
by SentinelOne

This is yet another post in which we'll show you how to get started with logging in a programming language or framework. Throughout the series, we've covered a lot of different languages and platforms: C#, Java, Ruby, Ruby On Rails, Python, and C++, just to name a few. Today's post will help you get started with Dart logging.

The post will loosely follow the same structure of the older posts in the series, but there might be som

overview of Dart. This makes sense if you consider that, despite being around for almost a decade and

Listen to this Post

Table of Contents
A Quick Overview on Dart
A Quick Overview on Dart
The Tip of the Dart Logging

https://www.sentinelone.com/blog/started-quickly-dart-logging/

universidade de aveiro

The END