The magic behind error tracking

How to build your own custom solution

Jonas Uekötter, 23. August 2022

About myself 👨‍💻

- I’m a Flutter developer since around Flutter 1.0
- Flutter Favorite Package Author: feedback
- Twitter: @ue_man
- GitHub @ueman

📱 💥 🕵️‍♂️

What’s in this presentation?

- We’re going to build the client side SDK for error monitoring
- We call it Sherlock
- What’s important in an error report?
- How to capture exceptions?
- How to fix common problems when using error monitoring tools?
- We’re ignoring the native part

What makes an error report actionable?

Exception & stack trace

- What's wrong?
- Where did it go wrong?
- Which stack frames are code from the app, which aren't?

```
class ErrorReport {
    final Object exception;
    final StackTrace stackTrace;
}
```

Caught vs Uncaught

vs crashed

- Did we catch the error?
- Was the error caught by a global error handler?
- We know, generally speaking, how bad a manually caught error is
- We don’t know how bad an uncaught error is, therefor we should look at them

```
class ErrorReport {
    final Object exception;
    final StackTrace stackTrace;
    final bool uncaught;
}
```

Environment information

Reproducibility

- Code might behave different on different devices
- iOS vs Android
- Phones vs Tablets

```
class ErrorReport {
    final Object exception;
    final StackTrace stackTrace;
    final bool uncaught;
    final Map<String, dynamic> envInfo;
}
```

App version information

Traceability

- In which builds did we introduce the error?
- Which builds don’t have that error anymore?

```
class ErrorReport {
    final Object exception;
    final StackTrace stackTrace;
    final bool uncaught;
    final Map<String, dynamic> envInfo;
    final Map<String, dynamic> appInfo;
}
```

What lead to this error?

Reproducibility

- What are the events which happened before this error?
- Http traffic
- File IO
- Navigation events
- Clicks on buttons & co

```
class ErrorReport {
    final Object exception;
    final StackTrace stackTrace;
    final bool uncaught;
    final Map<String, dynamic> envInfo;
    final Map<String, dynamic> appInfo;
    final List<LogEntry> log;
}
```

What makes an error report actionable?

Recap

- Exception & stack trace
- Caught vs uncaught
- Environment information
- App information
- Logs before an error

How to capture an exception?

Manual reporting

- Report error in a try-catch block

```
class Sherlock {
  static void capture(
    Object exception,
    StackTrace trace, {
    bool uncaught = false,
  }) {
    // ...
  }

  static void log(LogEntry logEntry) {
    // ...
  }
}
```

FlutterError.onError

- Widget layer errors
- Platform communication errors

```
FlutterError.onError = (details) {
  final exception = details.exception;
  final stackTrace = details.stack;
  Sherlock.captureError(
    exception,
    stackTrace,
    uncaught: true,
  );
};
```

PlatformDispatcher.onError

- Dart related errors
- Async context errors
    - Exceptions thrown in unawaited async methods
- Formerly done by runZonedGuarded

```
WidgetsFlutterBinding.instance
  .platformDispatcher.onError =
  (Object exception, StackTrace trace) {
    Sherlock.captureError(
      exception,
      trace,
      uncaught: true,
    );
    return false;
};
```

How to capture an exception?

Recap

- Try-catch: Manual reporting of errors
- FlutterError.onError
- PlatformDispatcher.onError (formerly runZonedGuarded)

Common problems

Common problems

Reporting errors multiple times

```
void doSomething() {
  try {
    throw Exception();
  } catch (e, s) {
    Sherlock.capture(e, s);
  }
}
```

```
void doEvenMore() {
  try {
    doSomething();
  } catch (e, s) {
    Sherlock.capture(e, s);
  }
}
```

Common problems

Reporting errors multiple times

- Errors are from time to time logged multiple times
  - try-catch-report-rethrow
- Keep list of last x reports, don’t throw if error is contained within it

Common problems

Reporting errors multiple times

```
class Deduplicator {
  final Queue<int> _exceptionsToDeduplicate = Queue<int>();
  final _duplicatesCount = 10;

  bool isDuplicate(ErrorReport e) {
    final exceptionHashCode = e.hashCode;

    if (_exceptionsToDeduplicate.contains(exceptionHashCode)) {
      return true;
    }

    _exceptionsToDeduplicate.add(exceptionHashCode);
    if (_exceptionsToDeduplicate.length > _duplicatesCount) {
      _exceptionsToDeduplicate.removeFirst();
    }
    return false;
  }
}
```

Common problems

Reporting errors which aren’t fixable or bad

- Reporting errors which aren’t fixable
  - Network exceptions
- User has to fix it, not possible inside SDK
  - We have to educate our users
- Adding a callback to our SDK to filter errors

Common problems

Reporting errors which aren’t fixable or bad

```
Sherlock.shouldReport = (ErrorReport e) {
  if(e.exception is NetworkException) {
    return false;
  }

  return true;
}
```

Common problems

Using error monitoring as replacement for logging/analytics

- Error monitoring is not logging
- Error monitoring is not analytics tracking
- User has to fix it, not possible inside SDK
    - We have to educate our users

Common problems

Wrapped exceptions

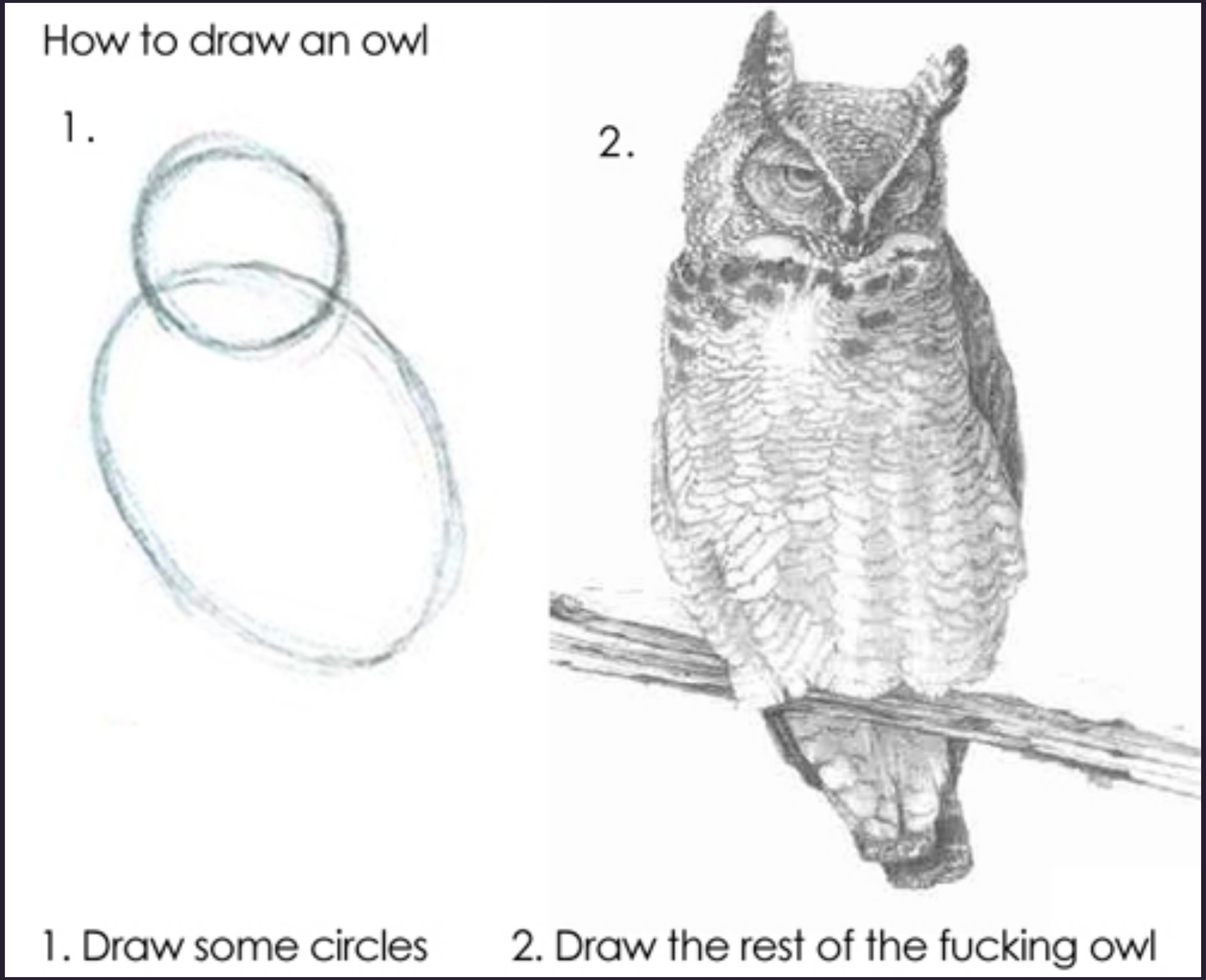
```
// Excerpt from https://pub.dev/packages/dio
// Popularity: 100
class DioError implements Exception {
  dynamic error;
  StackTrace? stackTrace;
}

// Excerpt from https://pub.dev/packages/gql_link
// Popularity: 93
abstract class LinkException implements Exception {
  final Object? originalException;
}
```

Recap: Common problems

- Reporting errors multiple times
- Reporting errors which aren’t fixable or bad
- Using error monitoring as replacement for logging or analytics
- Wrapped exceptions

Build the rest of the SDK



What’s missing?

- SDK needs to be initialized once to hook into global error handler
- Reports needs be serialized to json
- Serialized reports needs be sent to a server
- Write server code

What’s missing?

Initialize SDK

```
Sherlock.init(String inAppNamespace) {
  _saveInAppNameSpace();
  _hookIntoFlutterErrorOnError();
  _hookIntoPlatformDispatcherOnError();
}
```

What’s missing?

Serialize reports

```
class ErrorReport {
  // ...
  Map<String, dynamic> toJson() {
    // serialization code
  }
}
```

What’s missing?

Send error report

```
class Sherlock {
  static void capture(
    Object e, StackTrace trace,
    {bool uncaught = false}) {

    final report = _createReport(e, trace, uncaught);
    if(!shouldReport(report)) {
      return;
    }
    final json = report.toJson();
    _httpClient.post(json);
  }
}
```

What’s missing?

Write server

- Future talk
- Maybe with dart_frog?

👋

Sources & further reading

- https://master-api.flutter.dev/flutter/dart-ui/PlatformDispatcher/onError.html
- https://master-api.flutter.dev/flutter/foundation/FlutterError/onError.html
- https://pub.dev/packages/device_info_plus
- https://pub.dev/packages/package_info_plus
- https://pub.dev/packages/stack_trace
-