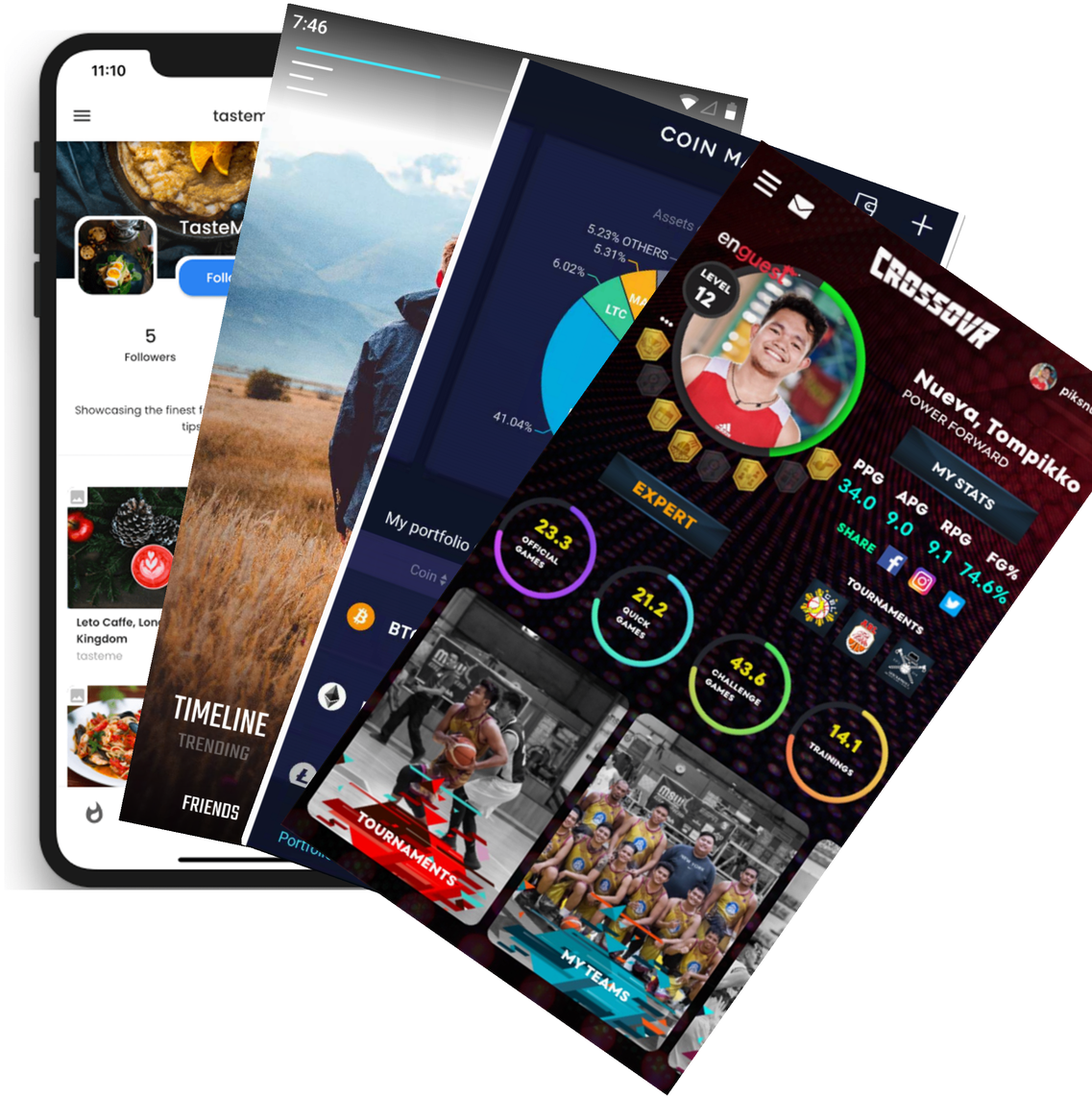
Flutter + Gomobile = 😍

Power up your mobile app with Go

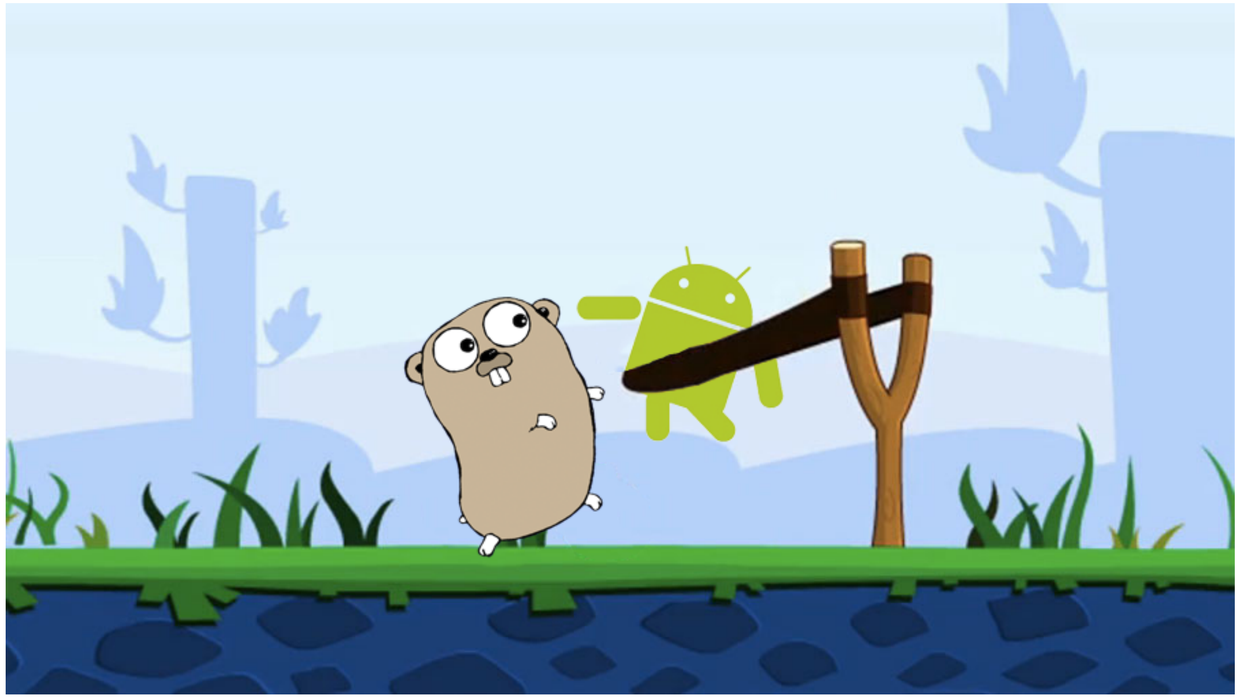
Ivan Danyliuk, @idanyliuk
GDG Barcelona Meetup
March 21, 2019

Flutter

- Brand new framework from Google for mobile apps
- Game changer in the mobile development field
- Unique and mindblowing design
- Production ready

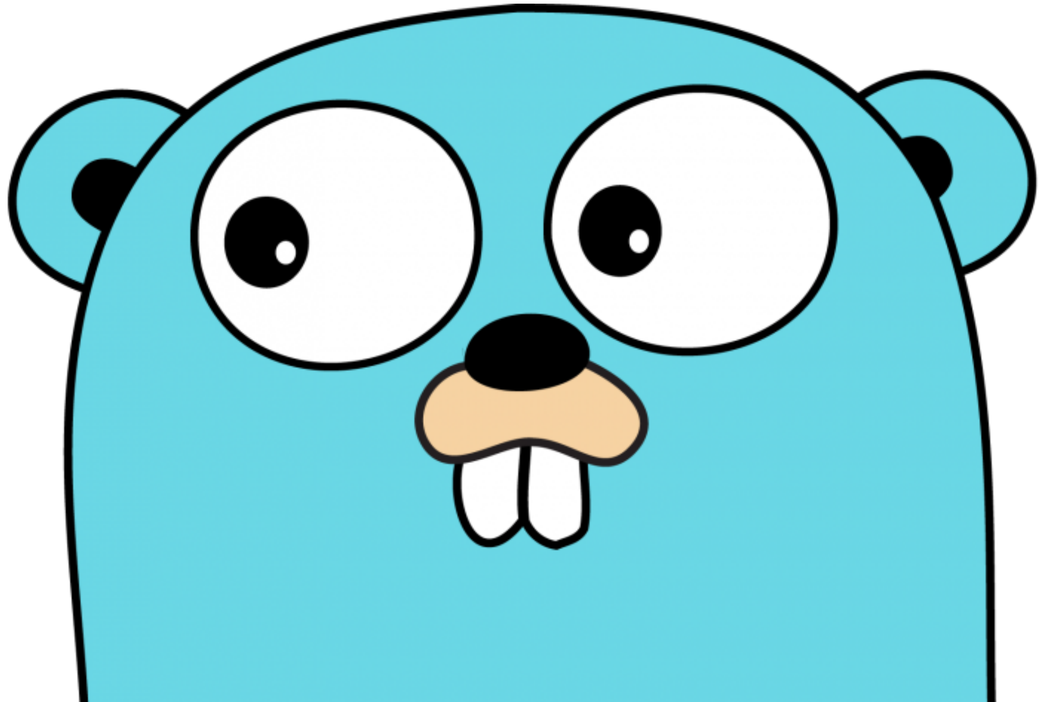


Gomobile



- Tooling for compiling Go code into native libraries for iOS and Android

- Open-source language developed in Google in 2009
- C of the XXI century
- Extremely simple and easy to learn
- Extremely productive
- Performant (on par with C++)
- Most of the cloud software is written in Go nowadays



Why you might need Go in Flutter?

Why you might need Go in Flutter?

- Flutter is written in Dart, which is relatively new language
    - Reusing existing Go code
    - Performance
    - Simpler language

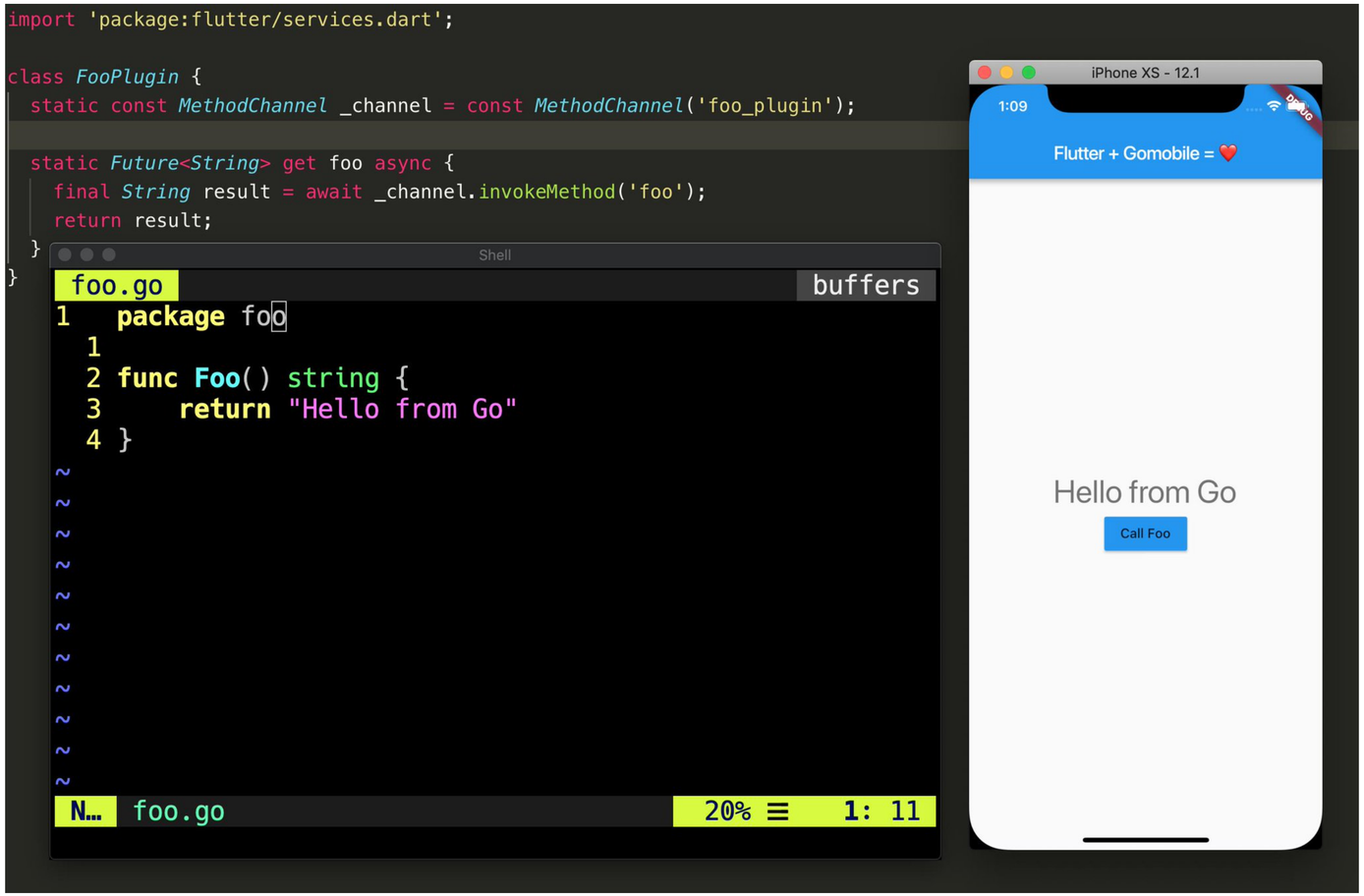
Reusing existing code

Reusing Go code

- Let's say, you want to create a mobile app running Ethereum light node
- Most advanced and most actively developed implementation of Ethereum is in Go - https://github.com/ethereum/go-ethereum
- ~400 contributors, ~770K LoC of Go, ~5 years of development
- Constantly changing and evolving specs

There is no way you can rewrite it in Dart.

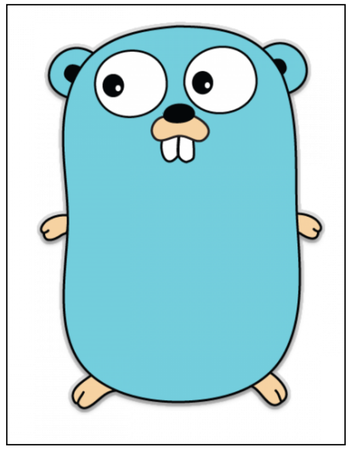
But you can reuse Go code using Gomobile in Flutter!



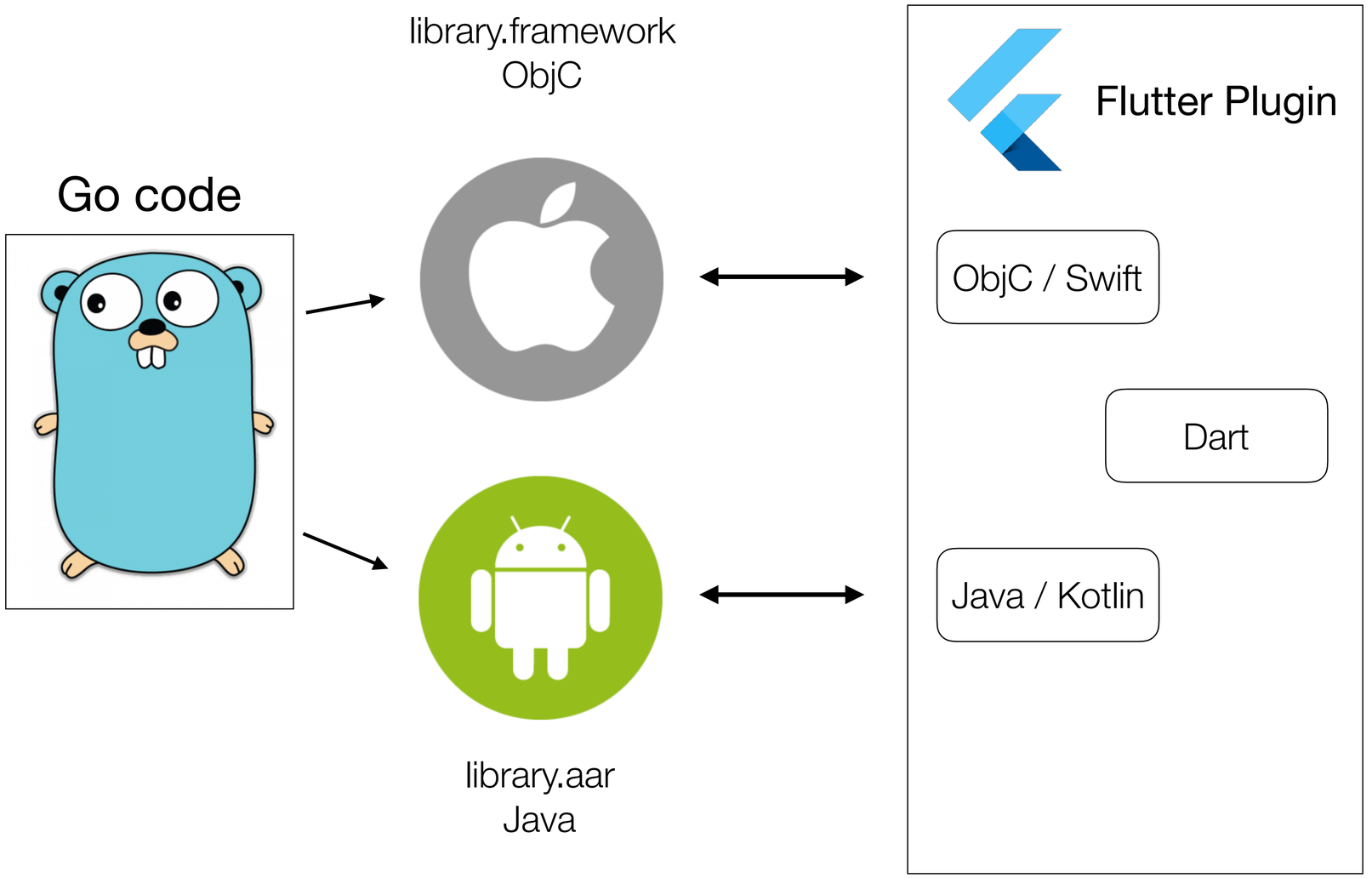
Flutter uses "platform channels" to talk native iOS/Android code.

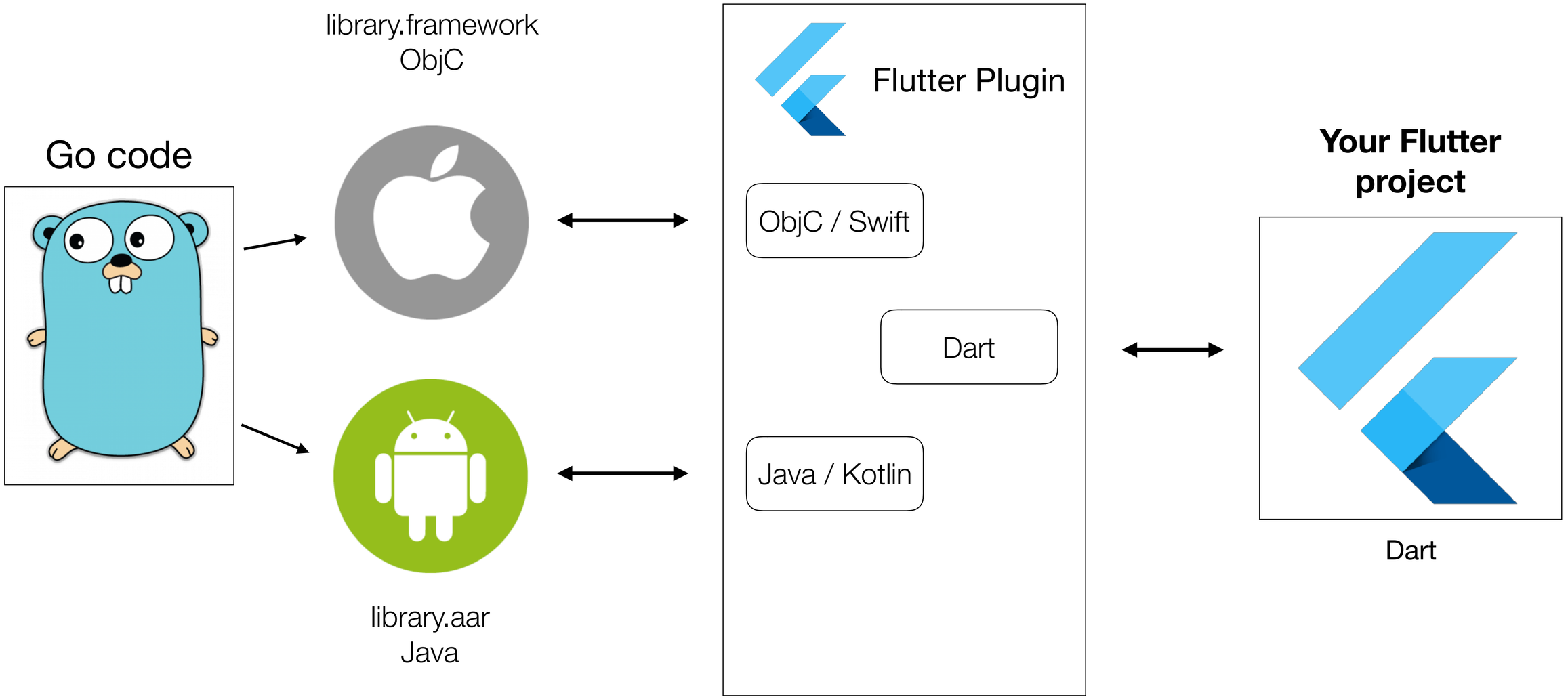
library.framework
ObjC

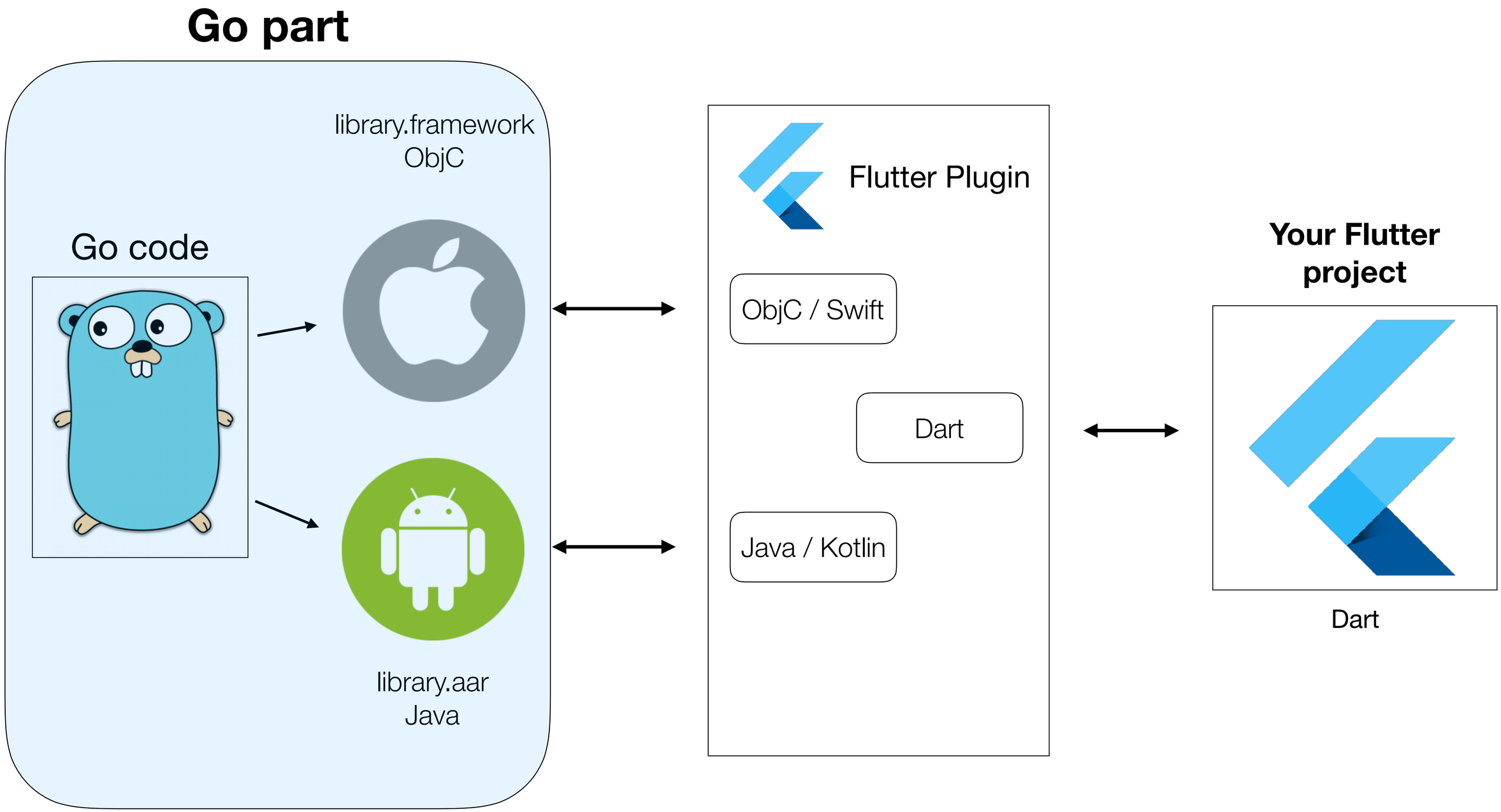
Go code

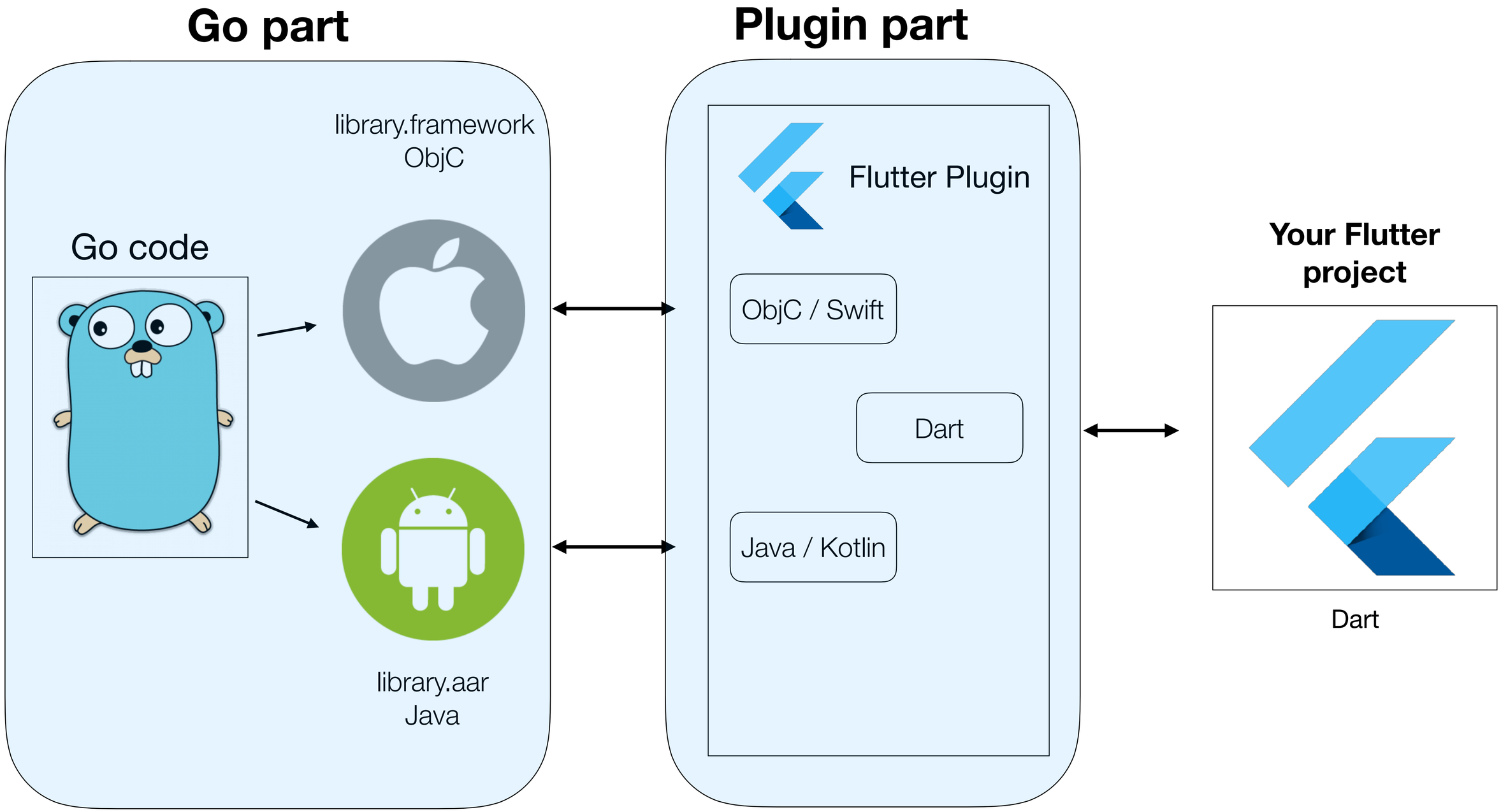


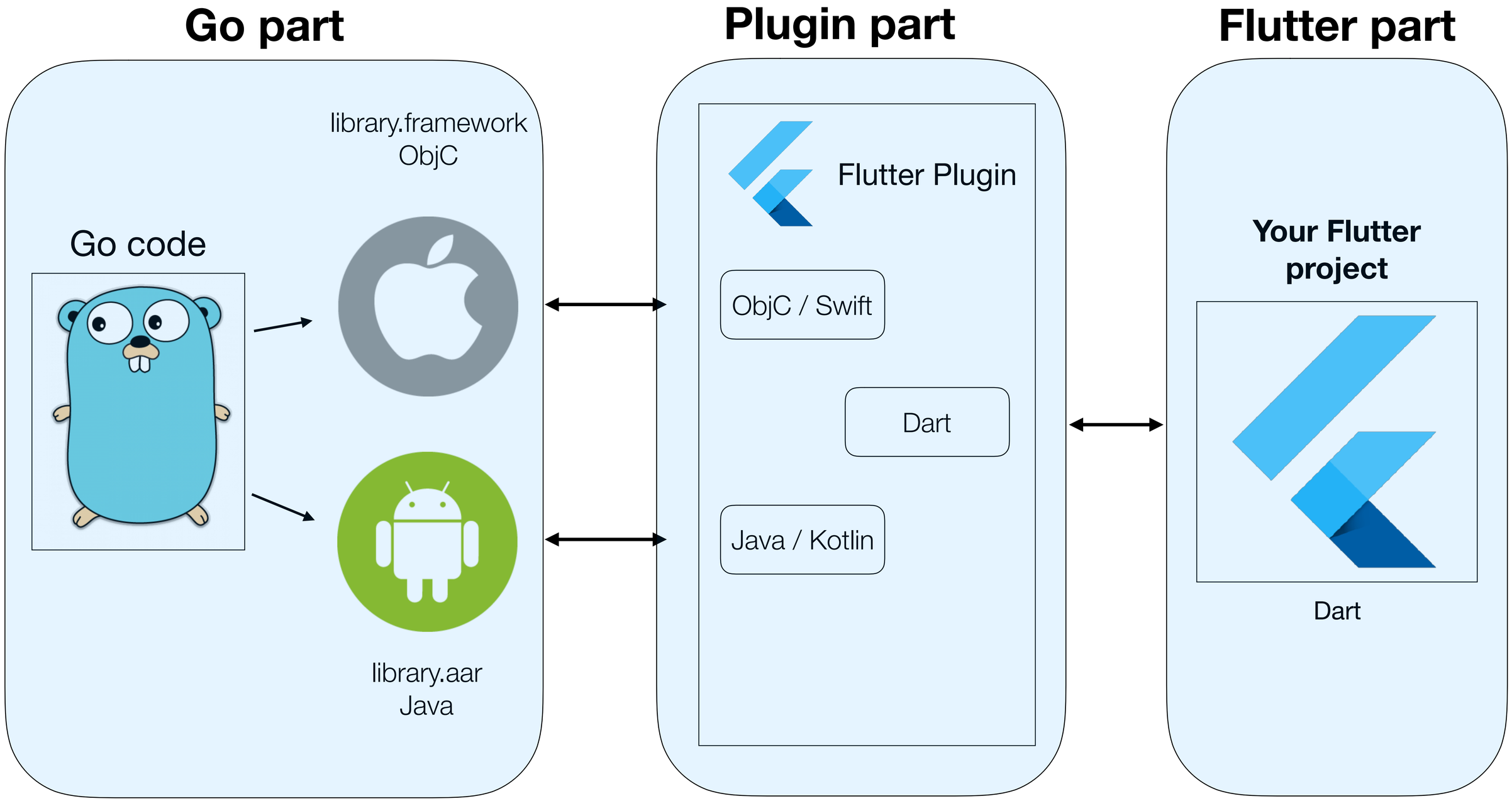
library.aar
Java


library.framework
ObjC
Go code
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
library.aar
Java

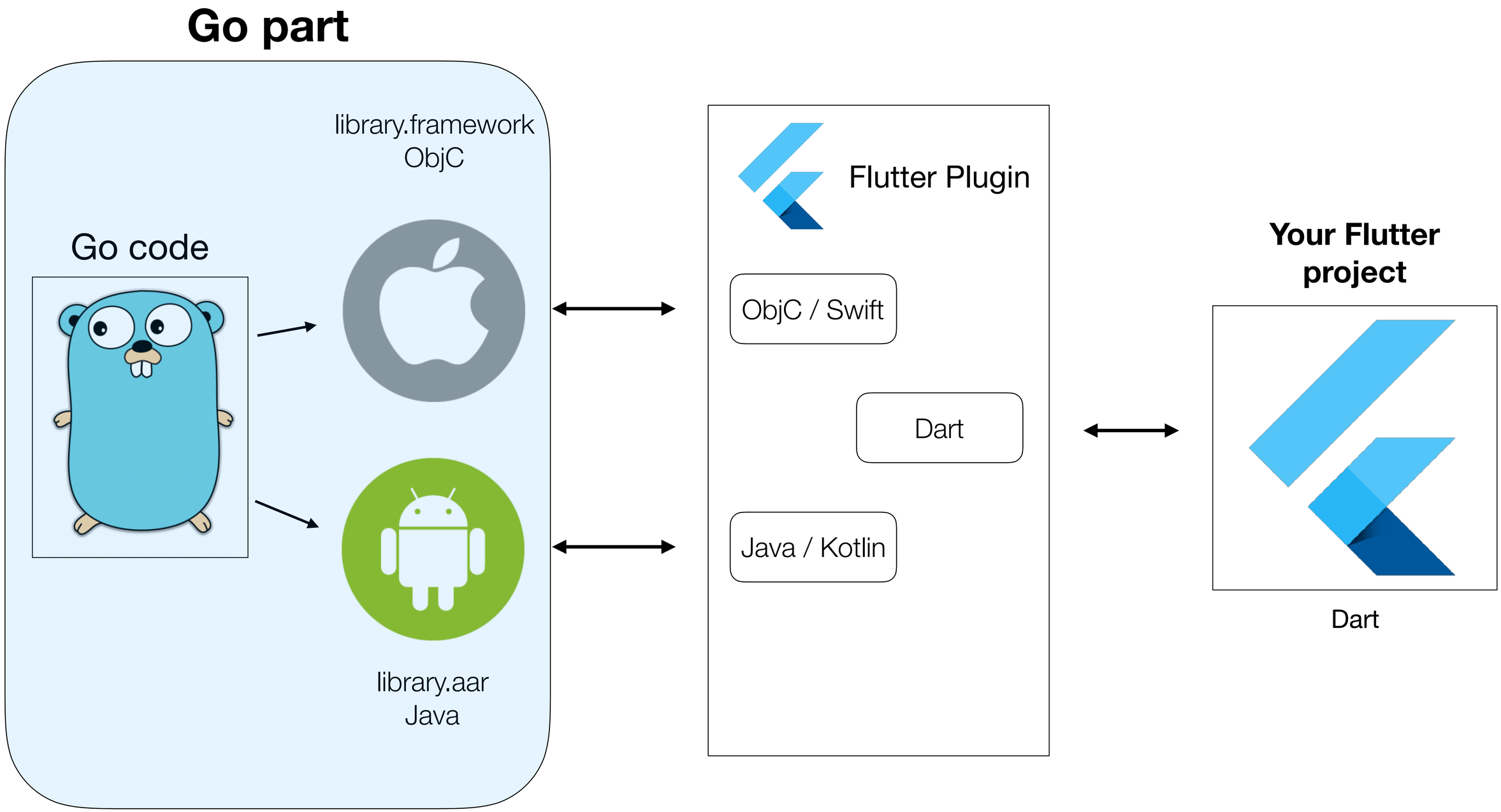

library.framework
ObjC
Go code
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
Your Flutter project
Dart
library.aar
Java


Go part
library.framework
ObjC
Go code
library.aar
Java
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
Your Flutter project
Dart


Go part
Plugin part
library.framework
ObjC
Go code
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
Your Flutter project
library.aar
Java
Dart


Go part
Plugin part
Flutter part
library.framework
ObjC
Go code
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
Your Flutter project
library.aar
Java
Dart

Go part


Go part
library.framework
ObjC
Go code
library.aar
Java
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
Your Flutter project
Dart

Example Go project

- Code in Go I wrote some years ago – github.com/divan/num2words
- Simple library for converting numbers into textual English representation
  - 42 → "fourty two"
  - 1111 → "one thousand one hundred and eleven"

num2words

build passing godoc reference

num2words - Numbers to words converter in Go (Golang)

Usage

First, import package num2words

```
import github.com/divan/num2words
```

Convert number

```
str := num2words.Convert(17) // outputs "seventeen"
...
str := num2words.Convert(1024) // outputs "one thousand twenty four"
...
str := num2words.Convert(-123) // outputs "minus one hundred twenty three"
```

Convert number with " and " between number groups:

```
str := num2words.ConvertAnd(514) // outputs "five hundred and fourteen"
...
str := num2words.ConvertAnd(123) // outputs "one hundred and twenty three"
```

Example Go project

Example Go project

```
$ time gomobile bind -target ios
real    0m19.134s
...

$ ls -w1
LICENSE
Num2words.framework
README.md
num2words.go
num2words_test.go
```

Example Go project

```
$ time gomobile bind -target ios
real    0m19.134s
...

$ ls -w1
LICENSE
Num2words.framework
README.md
num2words.go
num2words_test.go
```

Example Go project

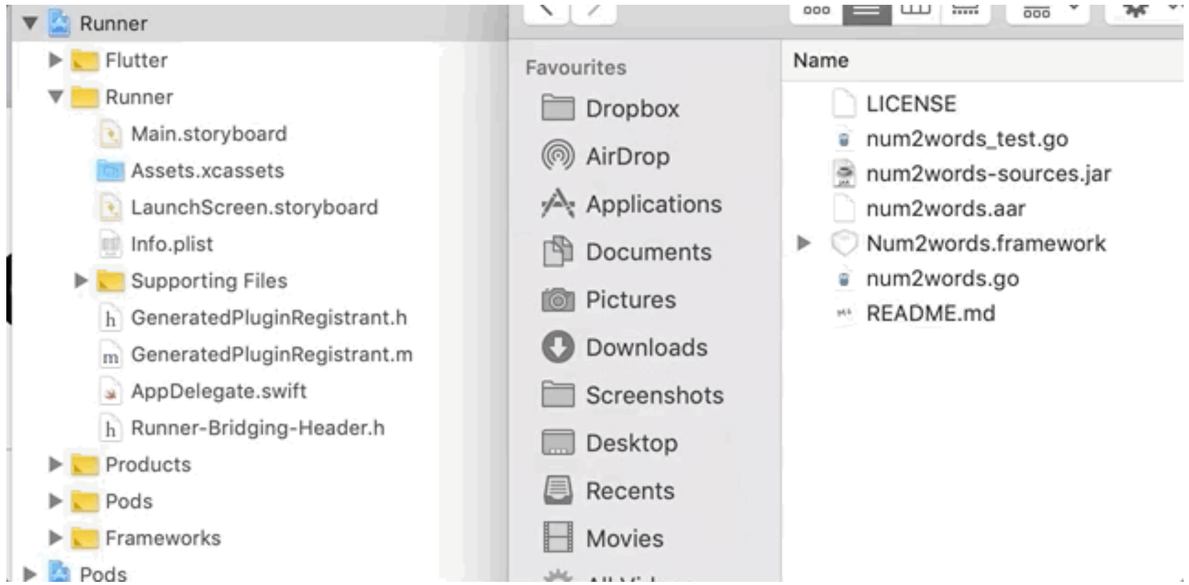
```
$ time gomobile bind -target android

real    0m36.399s
user    0m39.917s
sys 0m14.080s

$ ls -w1
LICENSE
Num2words.framework
README.md
num2words-sources.jar
num2words.aar
num2words.go
num2words_test.go
```

Example Go project

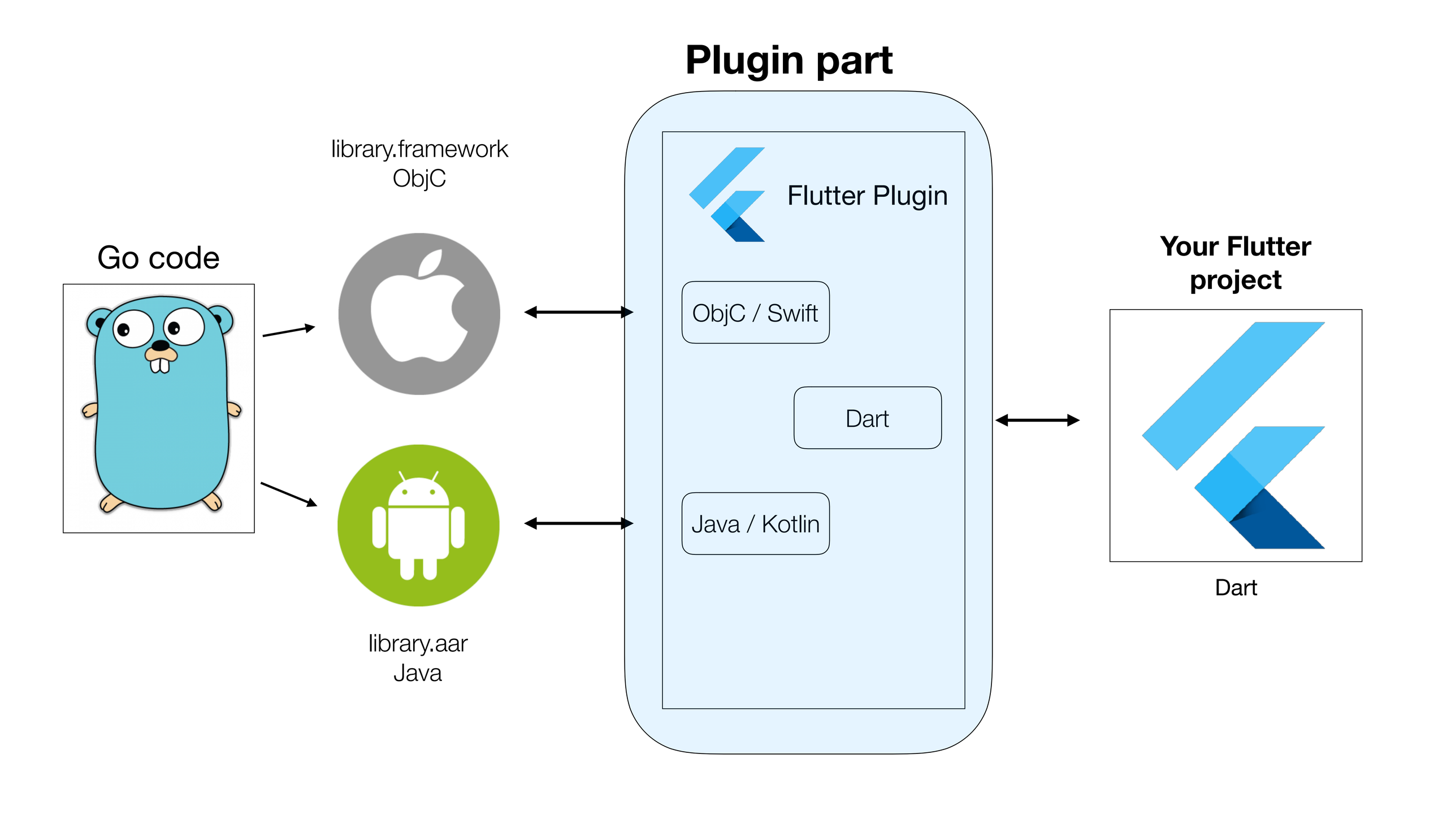
```
$ time gomobile bind -target android

real    0m36.399s
user    0m39.917s
sys 0m14.080s

$ ls -w1
LICENSE
Num2words.framework
README.md
num2words-sources.jar
num2words.aar
num2words.go
num2words_test.go
```

Gomobile Usage

- In a "normal" iOS or Android project:
  - Copy / Drag-and-Drop .aar and .framework files into native codebase
  - Import as a normal library



Flutter Plugin part


Plugin part
library.framework
ObjC
Go code
Flutter Plugin
ObjC / Swift
Dart
Java / Kotlin
Your Flutter project
library.aar
Java
Dart

Create plugin

```
$ flutter create --org com.divan --template=plugin -i swift -a kotlin
num2words_plugin
Creating project num2words_plugin...
  num2words_plugin/LICENSE (created)
  num2words_plugin/ios/num2words_plugin.podspec (created)
  num2words_plugin/ios/.gitignore (created)
  num2words_plugin/ios/Assets/.gitkeep (created)
  num2words_plugin/test/num2words_plugin_test.dart (created)
...
Running "flutter packages get" in example...                    4.9s
Wrote 92 files.

All done!
...
Your plugin code is in num2words_plugin/lib/num2words_plugin.dart.

Host platform code is in the "android" and "ios" directories under
num2words_plugin.
```

Create plugin

```
$ flutter create --org com.divan --template=plugin -i swift -a kotlin
num2words_plugin
Creating project num2words_plugin...
  num2words_plugin/LICENSE (created)
  num2words_plugin/ios/num2words_plugin.podspec (created)
  num2words_plugin/ios/.gitignore (created)
  num2words_plugin/ios/Assets/.gitkeep (created)
  num2words_plugin/test/num2words_plugin_test.dart (created)
...
Running "flutter packages get" in example...                    4.9s
Wrote 92 files.

All done!
...
Your plugin code is in num2words_plugin/lib/num2words_plugin.dart.

Host platform code is in the "android" and "ios" directories under
num2words_plugin.
```

Create plugin

```
$ flutter create --org com.divan --template=plugin -i swift -a kotlin
num2words_plugin
Creating project num2words_plugin...
  num2words_plugin/LICENSE (created)
  num2words_plugin/ios/num2words_plugin.podspec (created)
  num2words_plugin/ios/.gitignore (created)
  num2words_plugin/ios/Assets/.gitkeep (created)
  num2words_plugin/test/num2words_plugin_test.dart (created)
...
Running "flutter packages get" in example...                    4.9s
Wrote 92 files.

All done!
...
Your plugin code is in num2words_plugin/lib/num2words_plugin.dart.

Host platform code is in the "android" and "ios" directories under
num2words_plugin.
```

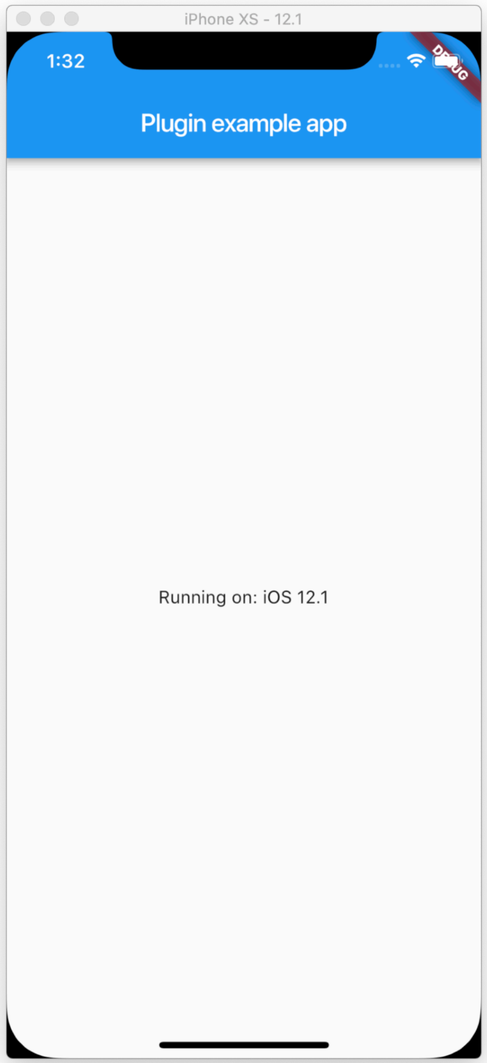
Create plugin

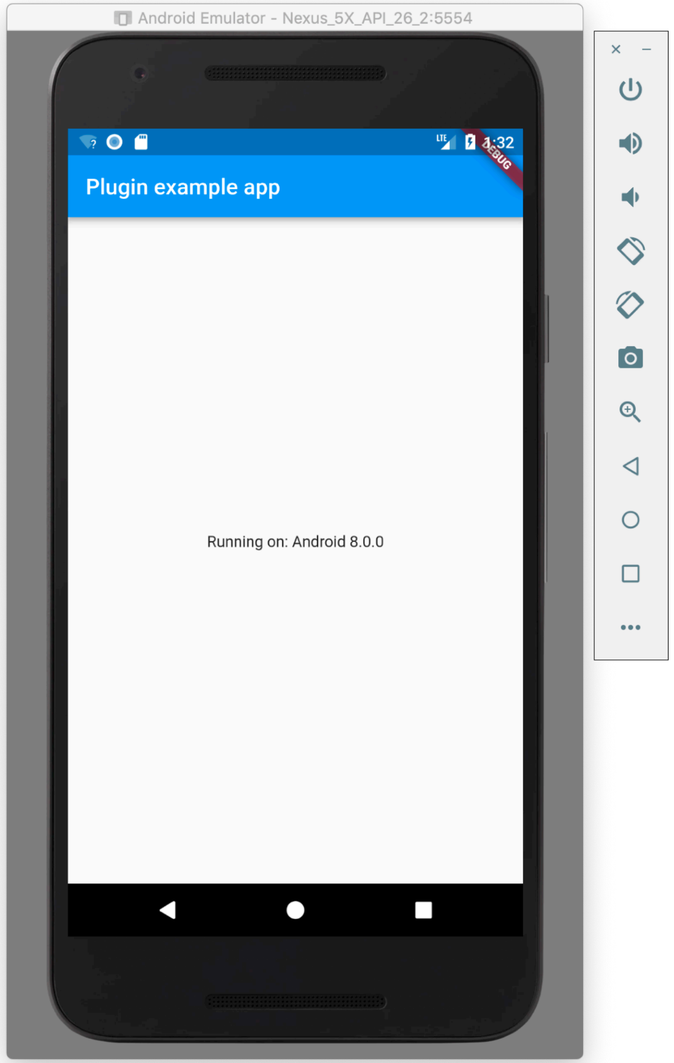
It contains an example/ app that shows you how to use the plugin. It has imlemented platformVersion method by default for example purposes.

```
$ cat example/lib/main.dart
import 'package:flutter/material.dart';
import 'dart:async';

import 'package:flutter/services.dart';
import 'package:num2words_plugin/num2words_plugin.dart';

...

    try {
      platformVersion = await Num2wordsPlugin.platformVersion;
    } on PlatformException {
      platformVersion = 'Failed to get platform version.';
    }
...
```

Create plugin

```
$ flutter run -d all
```





Connect plugin code and gomobile generated libraries

Flutter Plugin

- Let's see how plugin is structured
- Three source files in plugin:
  - **lib/num2words_plugin.dart** - plugin "public" interface
  - **ios/Classes/SwiftNum2wordsPlugin.swift** – iOS specific code
  - **android/src/main/kotlin/com/divan/num2words_plugin/Num2wordsPlugin.kt** - Android specific code
- We'll need to add our methods to **all three** files.

Flutter Plugin

- **lib/num2words_plugin.dart** - plugin "public" interface

```dart
import 'dart:async';

import 'package:flutter/services.dart';

class Num2wordsPlugin {
  static const MethodChannel _channel =
      const MethodChannel('num2words_plugin');

  static Future<String> get platformVersion async {
    final String version = await _channel.invokeMethod('getPlatformVersion');
    return version;
  }
}
```

Flutter Plugin

- **lib/num2words_plugin.dart** - plugin "public" interface

```dart
import 'dart:async';

import 'package:flutter/services.dart';

class Num2wordsPlugin {
  static const MethodChannel _channel =
      const MethodChannel('num2words_plugin');

  static Future<String> get platformVersion async {
    final String version = await _channel.invokeMethod('getPlatformVersion');
    return version;
  }

  static Future<String> convert (int num) async {
    final String version = await _channel.invokeMethod('convert', num);
    return version;
  }
}
```

Flutter Plugin

- **lib/num2words_plugin.dart** - plugin "public" interface

```dart
import 'dart:async';

import 'package:flutter/services.dart';

class Num2wordsPlugin {
  static const MethodChannel _channel =
      const MethodChannel('num2words_plugin');

  static Future<String> get platformVersion async {
    final String version = await _channel.invokeMethod('getPlatformVersion');
    return version;
  }

  static Future<String> convert (int num) async {
    final String version = await _channel.invokeMethod('convert', num);
    return version;
  }
}
```

iOS

Flutter Plugin

- **ios/Classes/SwiftNum2wordsPlugin.swift** – iOS specific code
- Copy Gomobile library to plugin ios/ folder

```
$ mkdir ios/Frameworks
$ cp -a ../num2words/Num2words.framework ios/Frameworks/
$ vim ios/num2words_plugin.podspec
# add a line:
s.ios.vendored_frameworks = 'Frameworks/Num2words.framework'
:wq
```

Flutter Plugin

- **ios/Classes/SwiftNum2wordsPlugin.swift** – iOS specific code

```swift
import Flutter
import UIKit

public class SwiftNum2wordsPlugin: NSObject, FlutterPlugin {
  public static func register(with registrar: FlutterPluginRegistrar) {
    let channel = FlutterMethodChannel(name: "num2words_plugin", binaryMessenger:
registrar.messenger())
    let instance = SwiftNum2wordsPlugin()
    registrar.addMethodCallDelegate(instance, channel: channel)
  }

  public func handle(_ call: FlutterMethodCall, result: @escaping FlutterResult) {
    result("iOS " + UIDevice.current.systemVersion)
  }
}
```

Flutter Plugin

- **ios/Classes/SwiftNum2wordsPlugin.swift** – iOS specific code

```swift
import Flutter
import UIKit

import Num2words

public class SwiftNum2wordsPlugin: NSObject, FlutterPlugin {
  public static func register(with registrar: FlutterPluginRegistrar) {
    let channel = FlutterMethodChannel(name: "num2words_plugin", binaryMessenger:
registrar.messenger())
    let instance = SwiftNum2wordsPlugin()
    registrar.addMethodCallDelegate(instance, channel: channel)
  }

  public func handle(_ call: FlutterMethodCall, result: @escaping FlutterResult) {
    switch call.method {
        case "convert":
            if let arg = call.arguments as? Int {
                result(Num2words.Num2wordsConvert(arg))
            } else {
                result(FlutterError.init(code: "BAD_ARGS", message: "Wrong argument
types", details: nil))
            }
        case "getPlatformVersion":
            result("iOS " + UIDevice.current.systemVersion)
        default:
            result(FlutterMethodNotImplemented)
    }
  }
}
```

Flutter Plugin

- **ios/Classes/SwiftNum2wordsPlugin.swift** – iOS specific code

```swift
import Flutter
import UIKit

import Num2words

public class SwiftNum2wordsPlugin: NSObject, FlutterPlugin {
  public static func register(with registrar: FlutterPluginRegistrar) {
    let channel = FlutterMethodChannel(name: "num2words_plugin", binaryMessenger:
registrar.messenger())
    let instance = SwiftNum2wordsPlugin()
    registrar.addMethodCallDelegate(instance, channel: channel)
  }

  public func handle(_ call: FlutterMethodCall, result: @escaping FlutterResult) {
    switch call.method {
        case "convert":
            if let arg = call.arguments as? Int {
                result(Num2words.Num2wordsConvert(arg))
            } else {
                result(FlutterError.init(code: "BAD_ARGS", message: "Wrong argument
types", details: nil))
            }
        case "getPlatformVersion":
            result("iOS " + UIDevice.current.systemVersion)
        default:
            result(FlutterMethodNotImplemented)
    }
  }
}
```

Flutter Plugin

- Next step: make XCode apply changes and rebuild
  - Open in Xcode and build there, or
  - run `**flutter build ios**` in your example/ project

Android

Flutter Plugin

- **android/src/main/kotlin/com/divan/num2words_plugin/Num2wordsPlugin.kt** – Android specific code

```kotlin
package com.divan.num2words_plugin

import io.flutter.plugin.common.MethodCall
import io.flutter.plugin.common.MethodChannel
import io.flutter.plugin.common.MethodChannel.MethodCallHandler
import io.flutter.plugin.common.MethodChannel.Result
import io.flutter.plugin.common.PluginRegistry.Registrar

import num2words.Num2words

class Num2wordsPlugin: MethodCallHandler {
  companion object {
    @JvmStatic
    fun registerWith(registrar: Registrar) {
      val channel = MethodChannel(registrar.messenger(), "num2words_plugin")
      channel.setMethodCallHandler(Num2wordsPlugin())
    }
  }

  override fun onMethodCall(call: MethodCall, result: Result) {
    if (call.method == "getPlatformVersion") {
      result.success("Android ${android.os.Build.VERSION.RELEASE}")
    } else if (call.method == "convert") {
      result.success(Num2words.convert(call.arguments));
    } else {
      result.notImplemented()
    }
  }
}
```

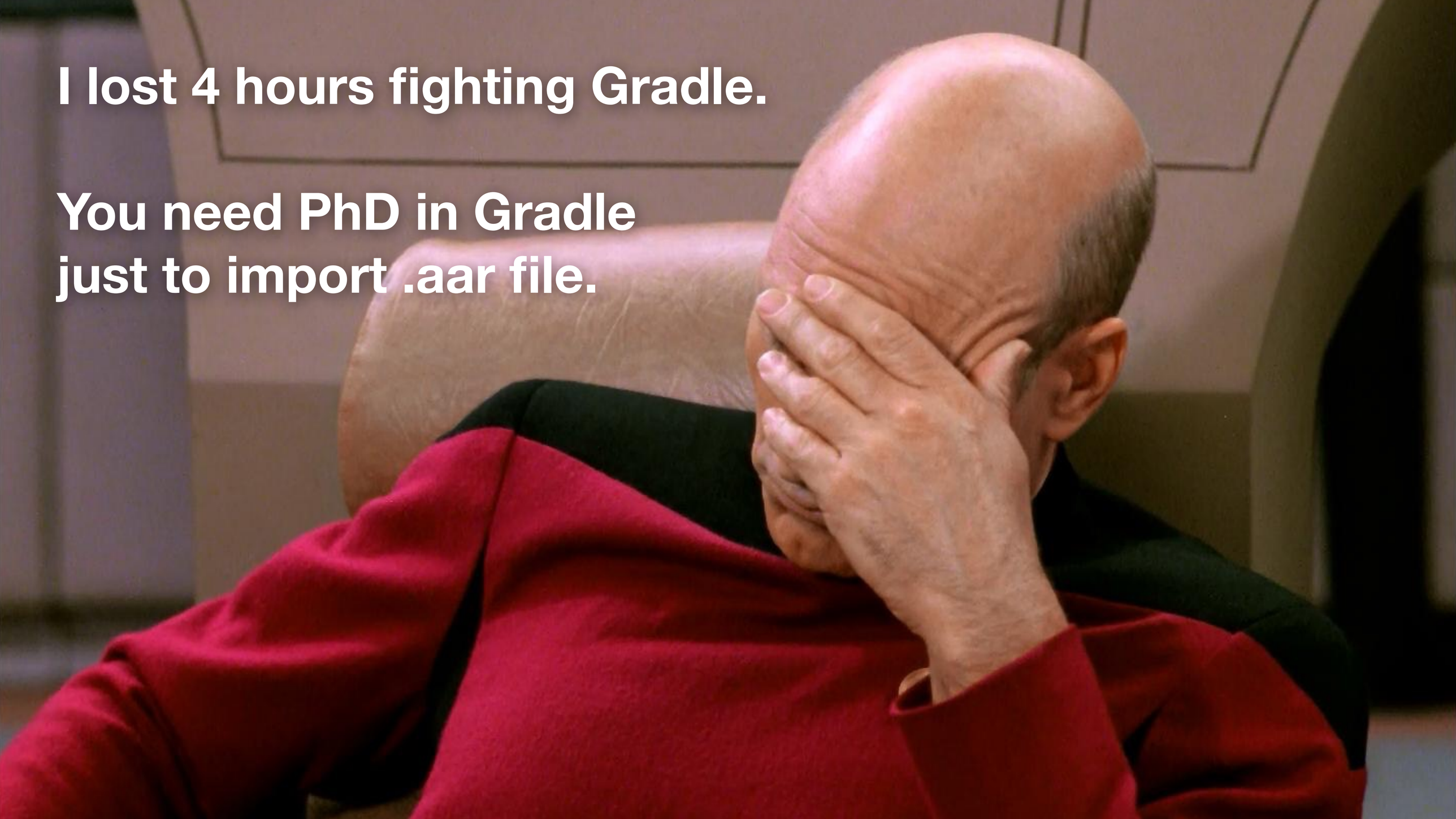
Flutter Plugin

- **android/src/main/kotlin/com/divan/num2words_plugin/Num2wordsPlugin.kt** – Android specific code

```kotlin
package com.divan.num2words_plugin

import io.flutter.plugin.common.MethodCall
import io.flutter.plugin.common.MethodChannel
import io.flutter.plugin.common.MethodChannel.MethodCallHandler
import io.flutter.plugin.common.MethodChannel.Result
import io.flutter.plugin.common.PluginRegistry.Registrar

import num2words.Num2words

class Num2wordsPlugin: MethodCallHandler {
  companion object {
    @JvmStatic
    fun registerWith(registrar: Registrar) {
      val channel = MethodChannel(registrar.messenger(), "num2words_plugin")
      channel.setMethodCallHandler(Num2wordsPlugin())
    }
  }

  override fun onMethodCall(call: MethodCall, result: Result) {
    if (call.method == "getPlatformVersion") {
      result.success("Android ${android.os.Build.VERSION.RELEASE}")
    } else if (call.method == "convert") {
      result.success(Num2words.convert(call.arguments));
    } else {
      result.notImplemented()
    }
  }
}
```

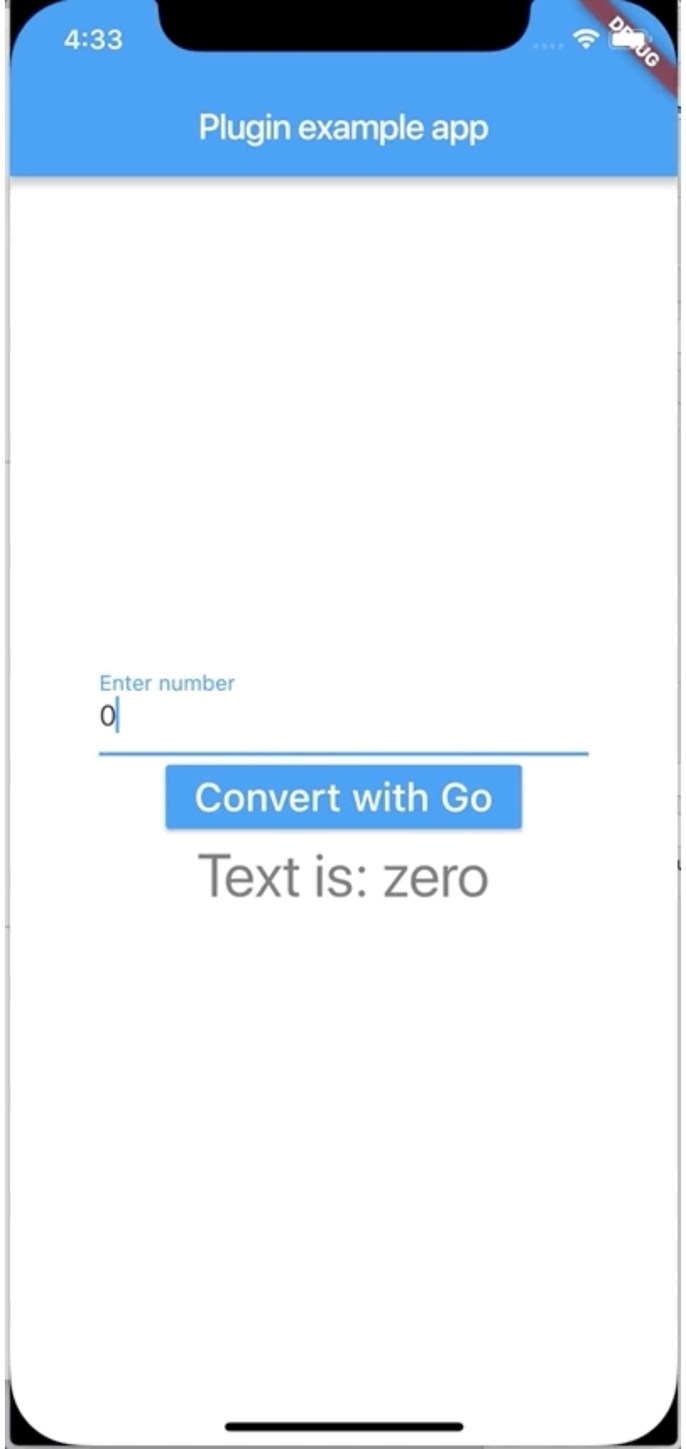
Flutter Plugin

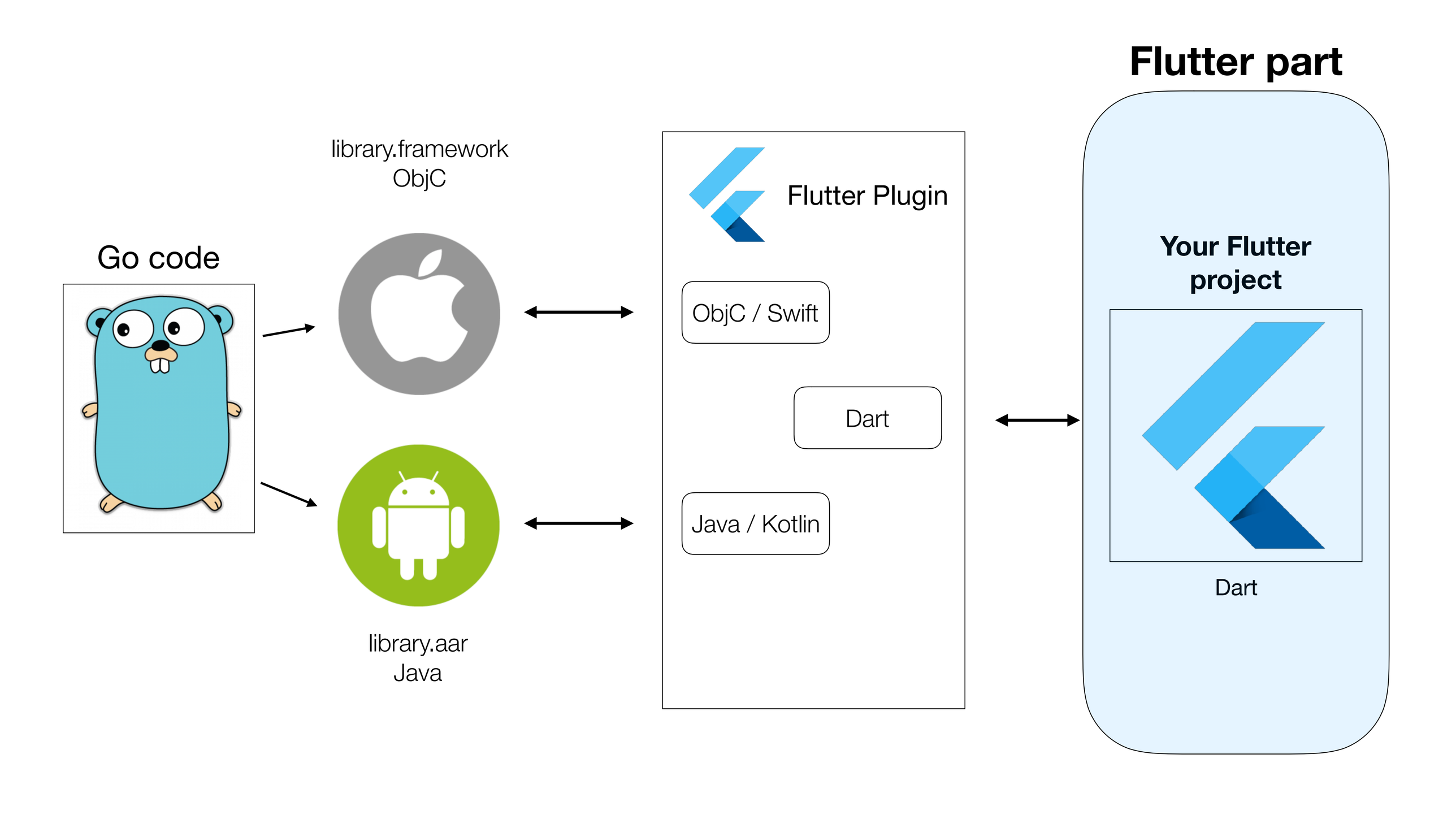
- To add gomobile .aar file to Android project
  - use Android studio, **File → New → New → Module → AAR/JAR file**
  - Manually add gradle/setting file if you're gradle wizard


I lost 4 hours fighting Gradle.
You need PhD in Gradle
just to import .aar file.

Running our example

(on iOS only, because Gradle)


4:33
Plugin example app
Enter number
0
Convert with Go
Text is: zero


Flutter part
library.framework
ObjC
Go code
Flutter Plugin
Your Flutter
project
ObjC / Swift
Dart
Java / Kotlin
Dart
library.aar
Java

Use plugin in app

pubspec.yaml

```yaml
dev_dependencies:
  flutter_test:
    sdk: flutter

  num2words:
    path: ../num2words_plugin
```

main.dart

```dart
import 'package:num2words_plugin/num2words_plugin.dart';

...
  // get value to convert
  Num2wordsPlugin.convert(value).then((str) {
   setState(() {
      text = str;
   });
  });
```

Conclusions

- Flutter + Gomobile is a powerful combo
- Keep "glue" API as simple as possible, because of type conversion nightmare
- No slices or maps in Gomobile
- If you need to pass complex data, it's probably easier just to encode it as a string

Links

- Flutter: Writing custom platform-specific code
- Flutter: Developing packages & plugins
- Building a mobile frontend for a Go application using Flutter
- Gomobile: Building libraries for SDK apps
- https://github.com/divan/num2words
- https://flutter.dev

Thank you

@idanyliuk