The world of Dart only plugins

Or: Accessing platform features without having to use method channels

renan.gg

Hello there 👋

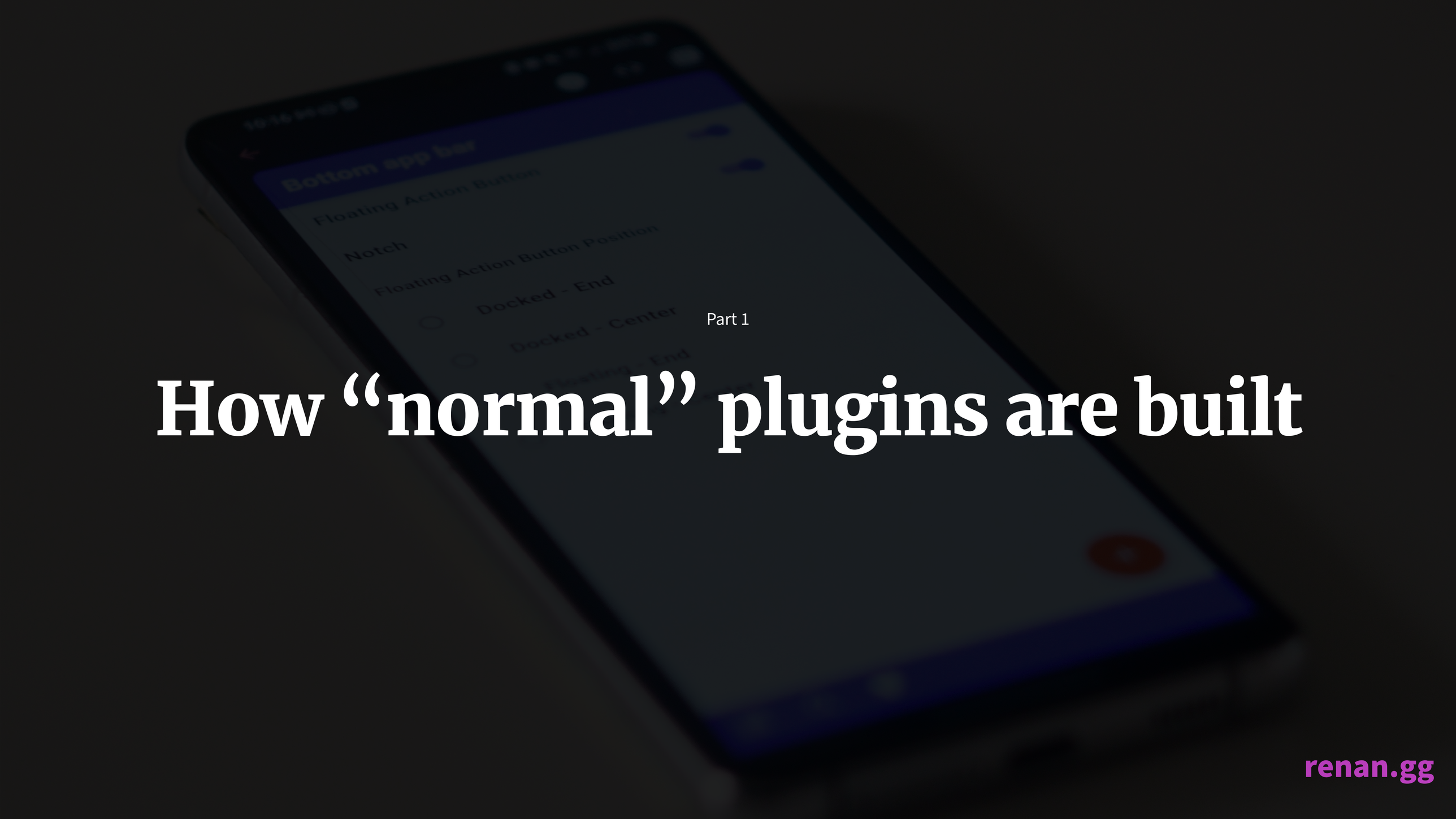
My name is

Renan Araújo

@reNotANumber

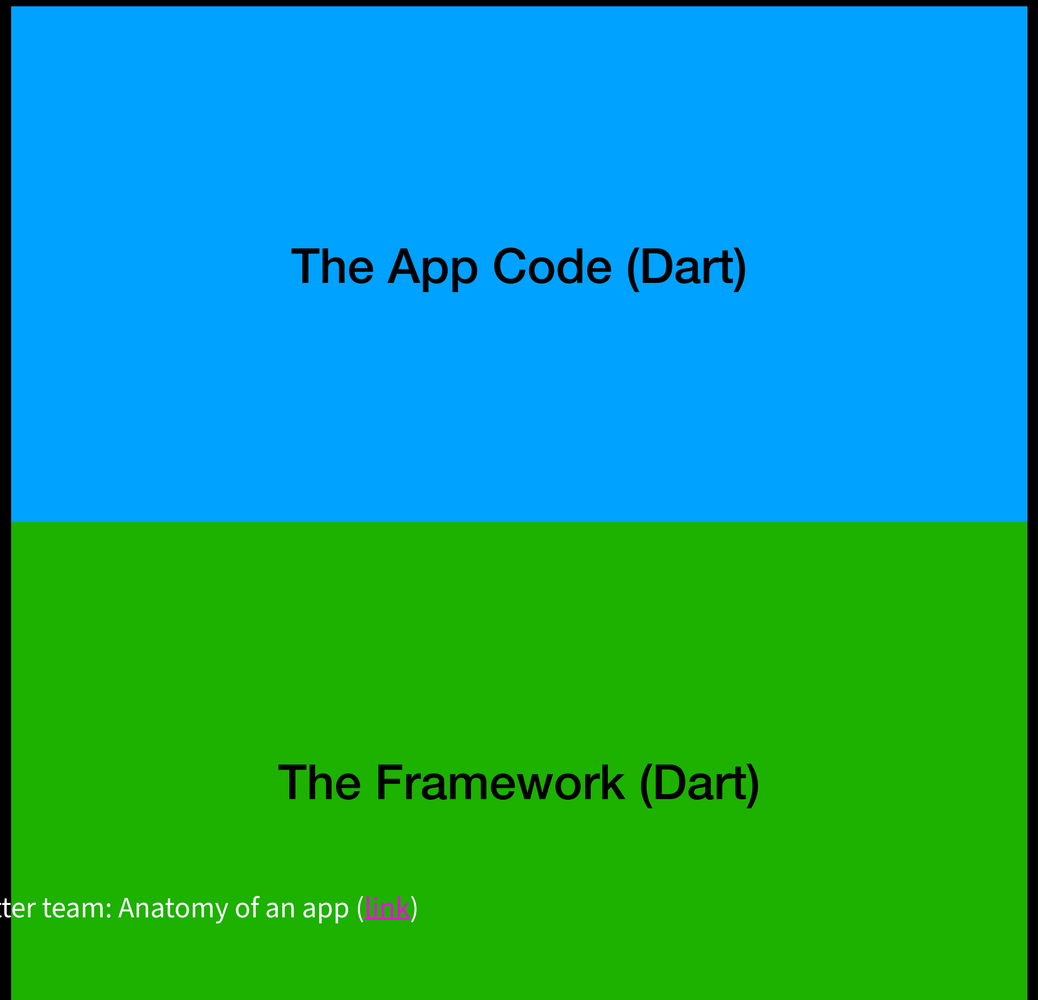
I live in **Porto**,
OSS Contributor for **Blue Fire** & OSS Engineer at **Very Good Ventures,**
You can see my socials and stuff at **renan.gg**

A new way to access native features



Part 1

How “normal” plugins are built



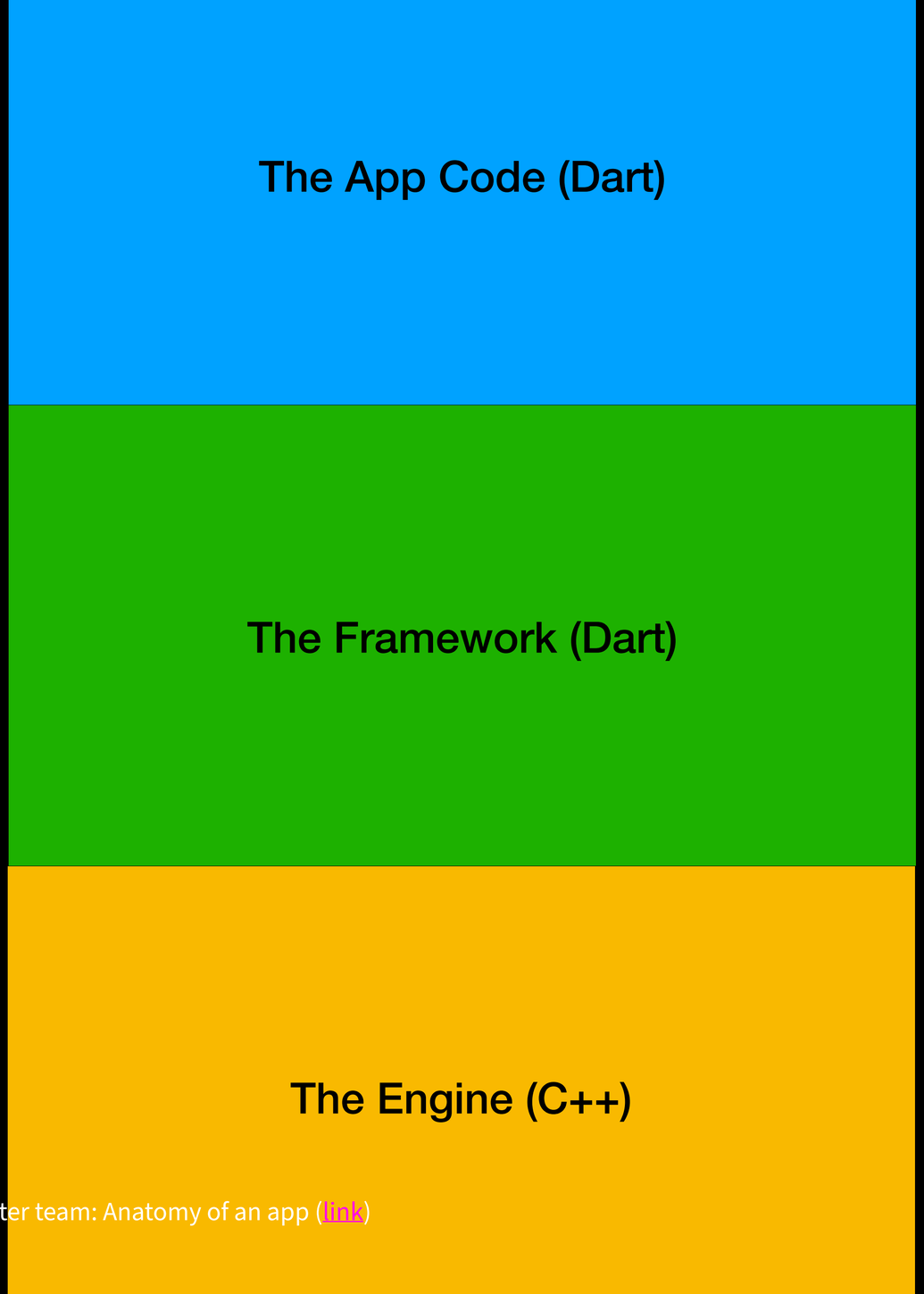
Everything we write:

All widgets we build,

Routes, State managers,

Packages…

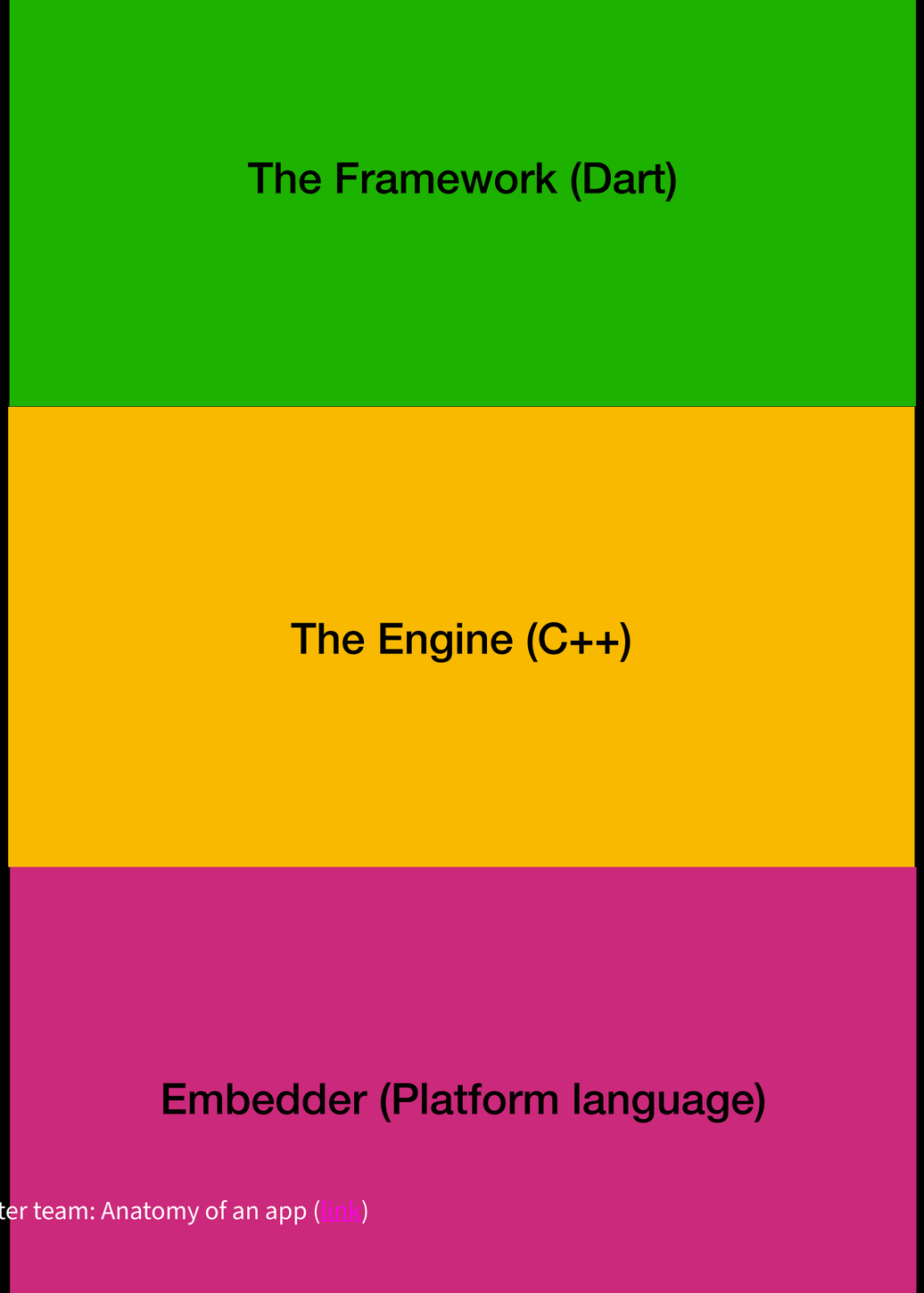
Credits to the Flutter team: Anatomy of an app (link)



Most of the API we interact with:

Material, Cupertino, Widgets.dart, etc

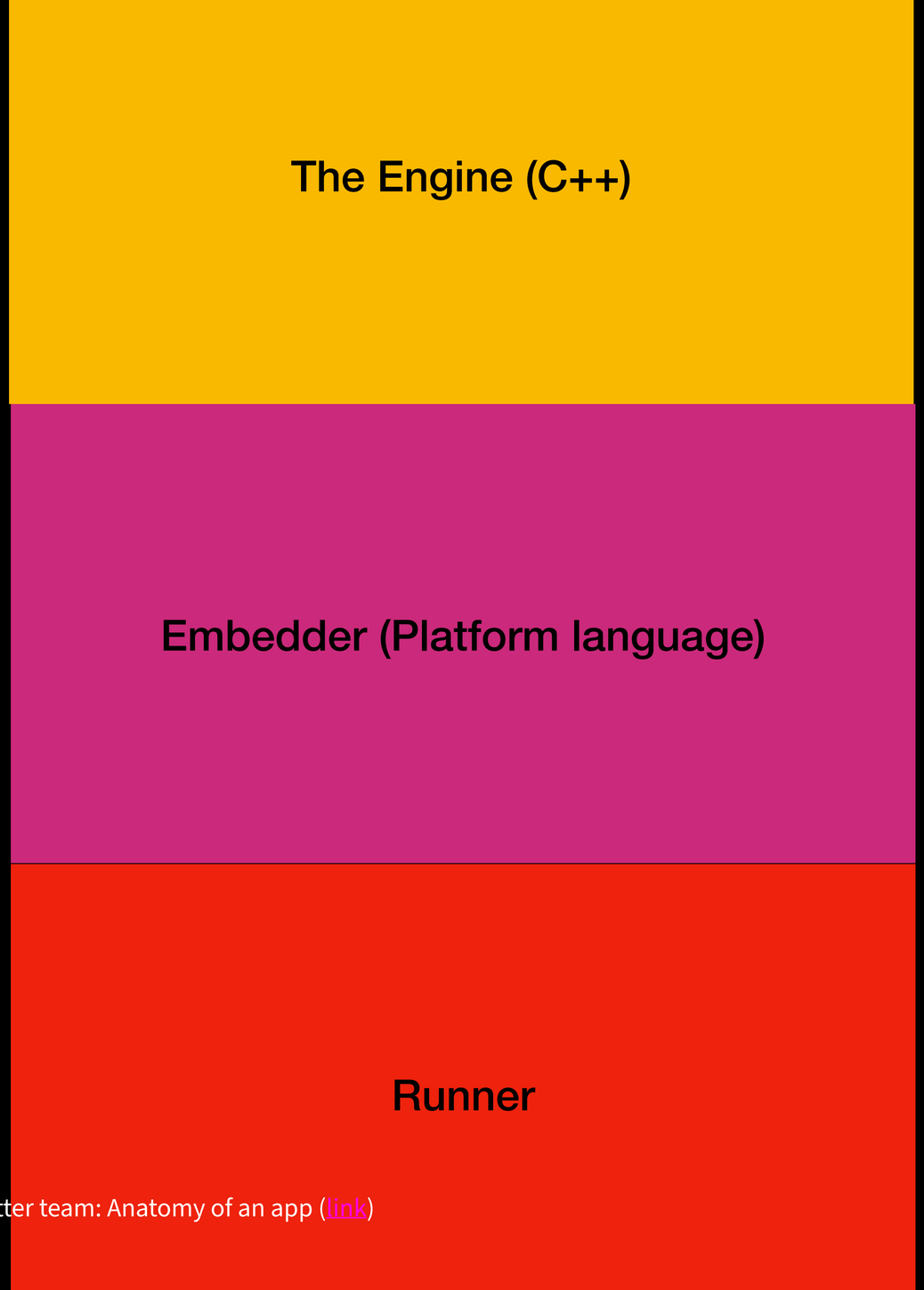
Credits to the Flutter team: Anatomy of an app (link)



Flutter Engine:

dart:ui, core APIs, embedder API

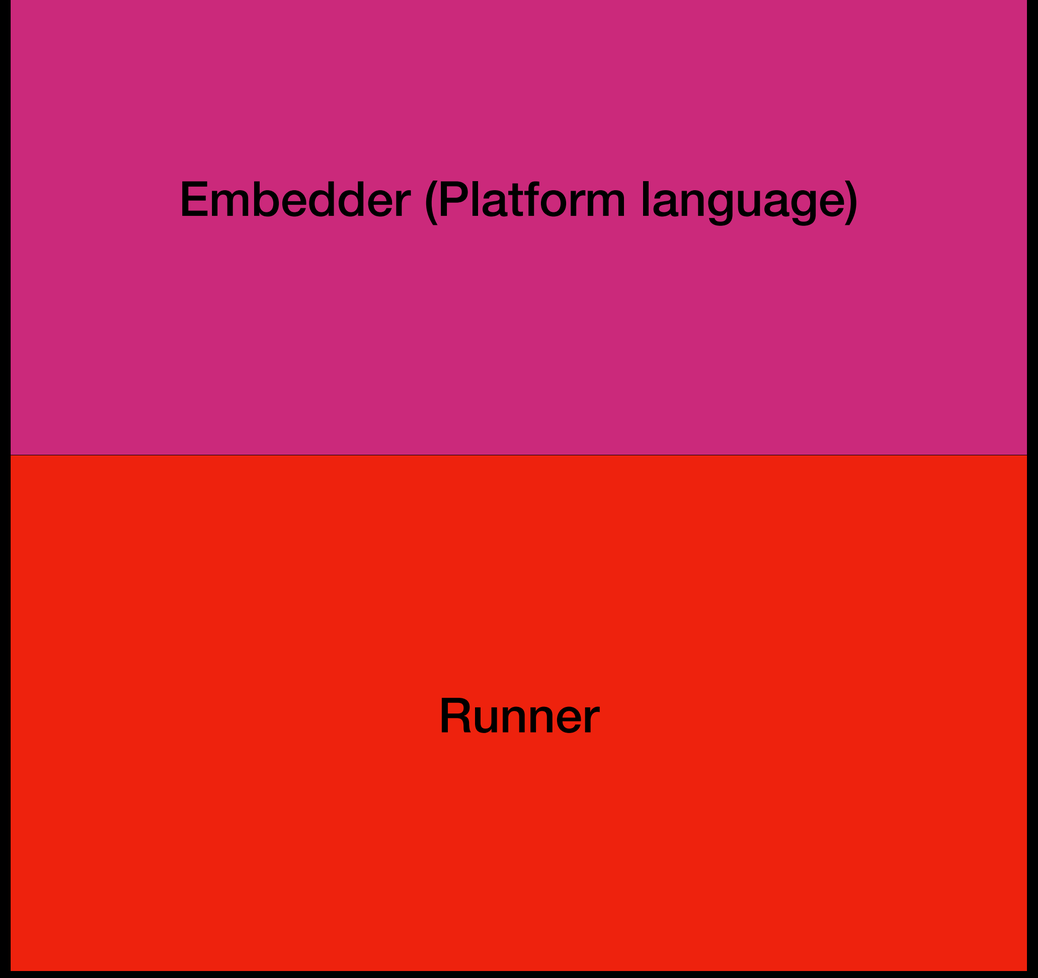
Credits to the Flutter team: Anatomy of an app (link)



Embedder:

Platform-specific code, sync the event loop with the platform, Access to platform APIs such as gestures, keyboard, etc

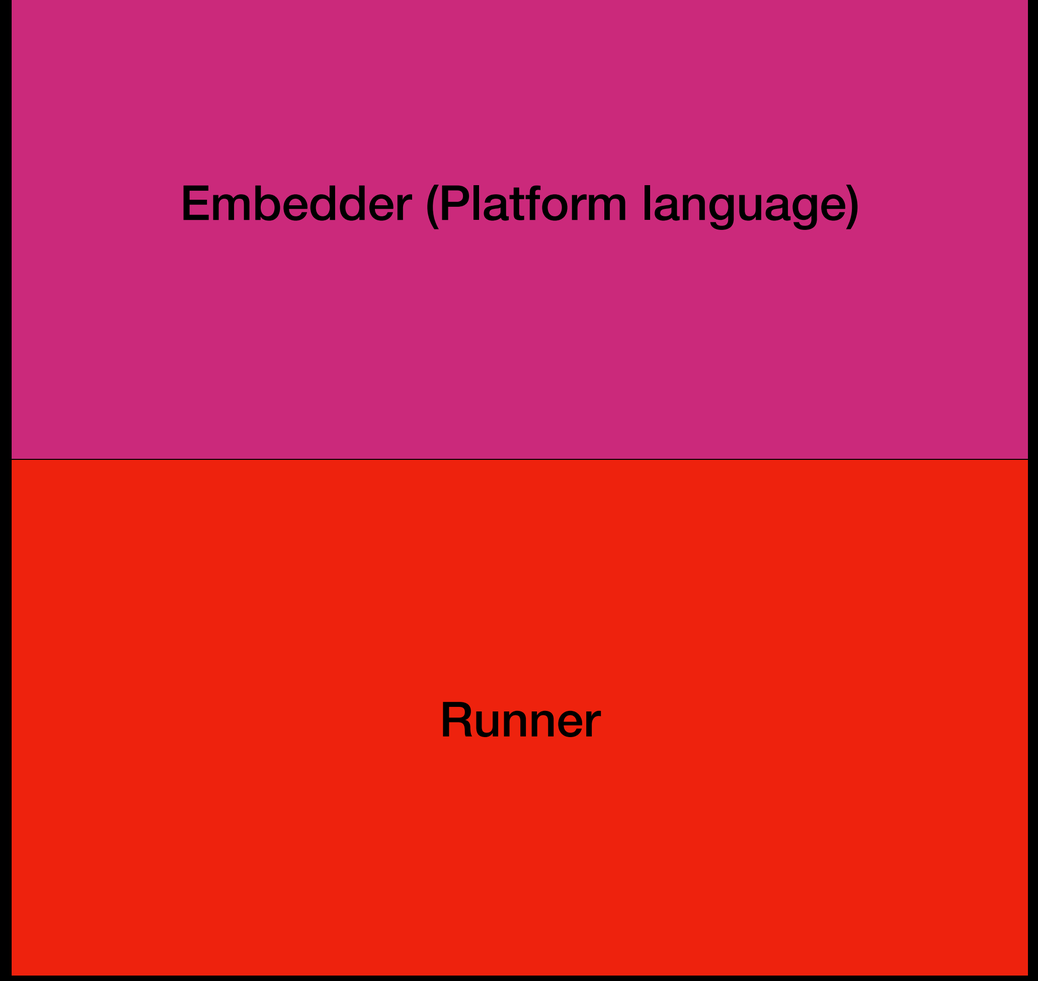
Credits to the Flutter team: Anatomy of an app (link)



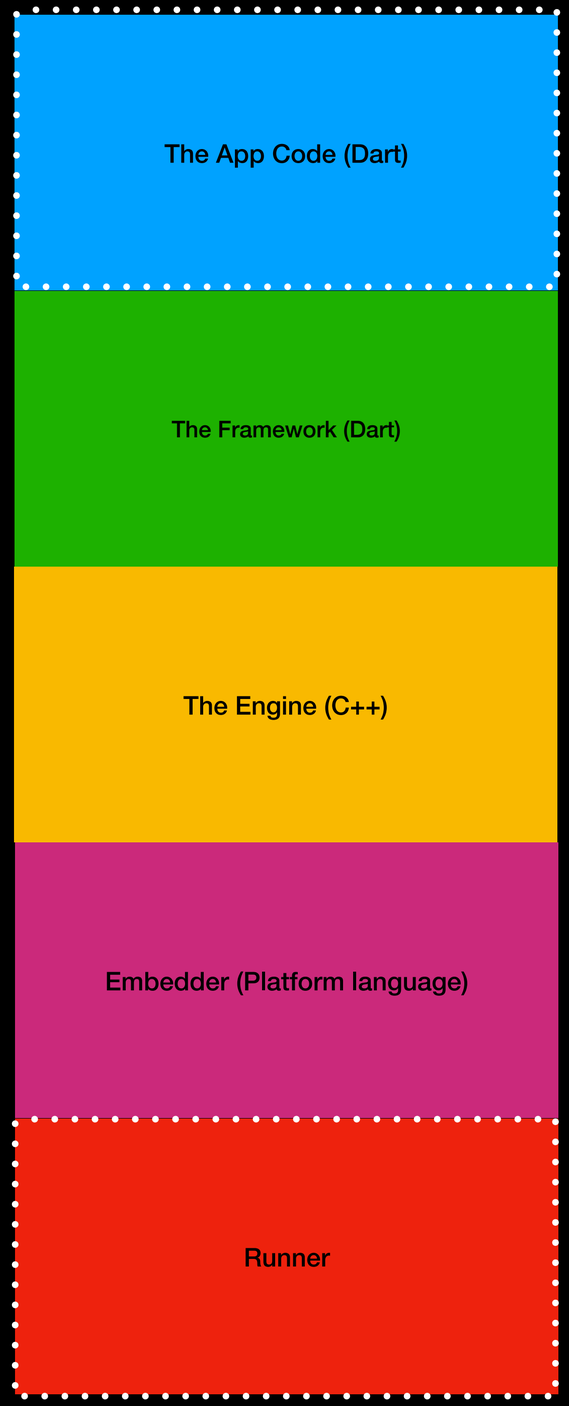
The Runner:

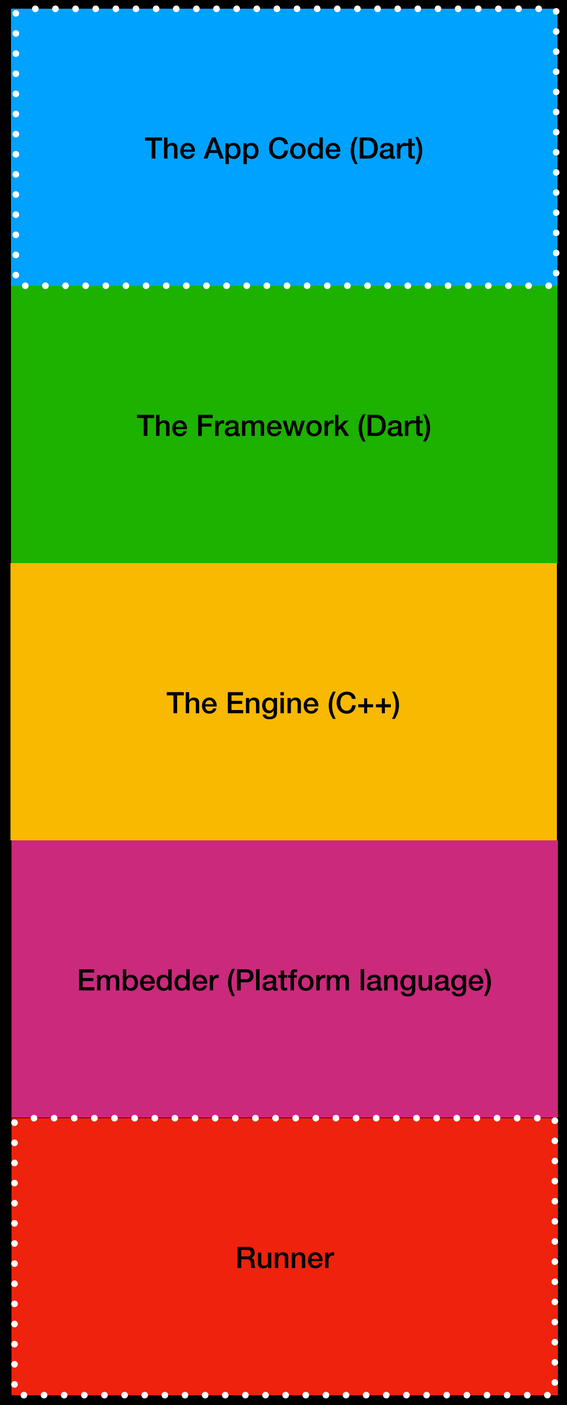
The actual native app that we gave under the platform directories: macos, ios, windows…

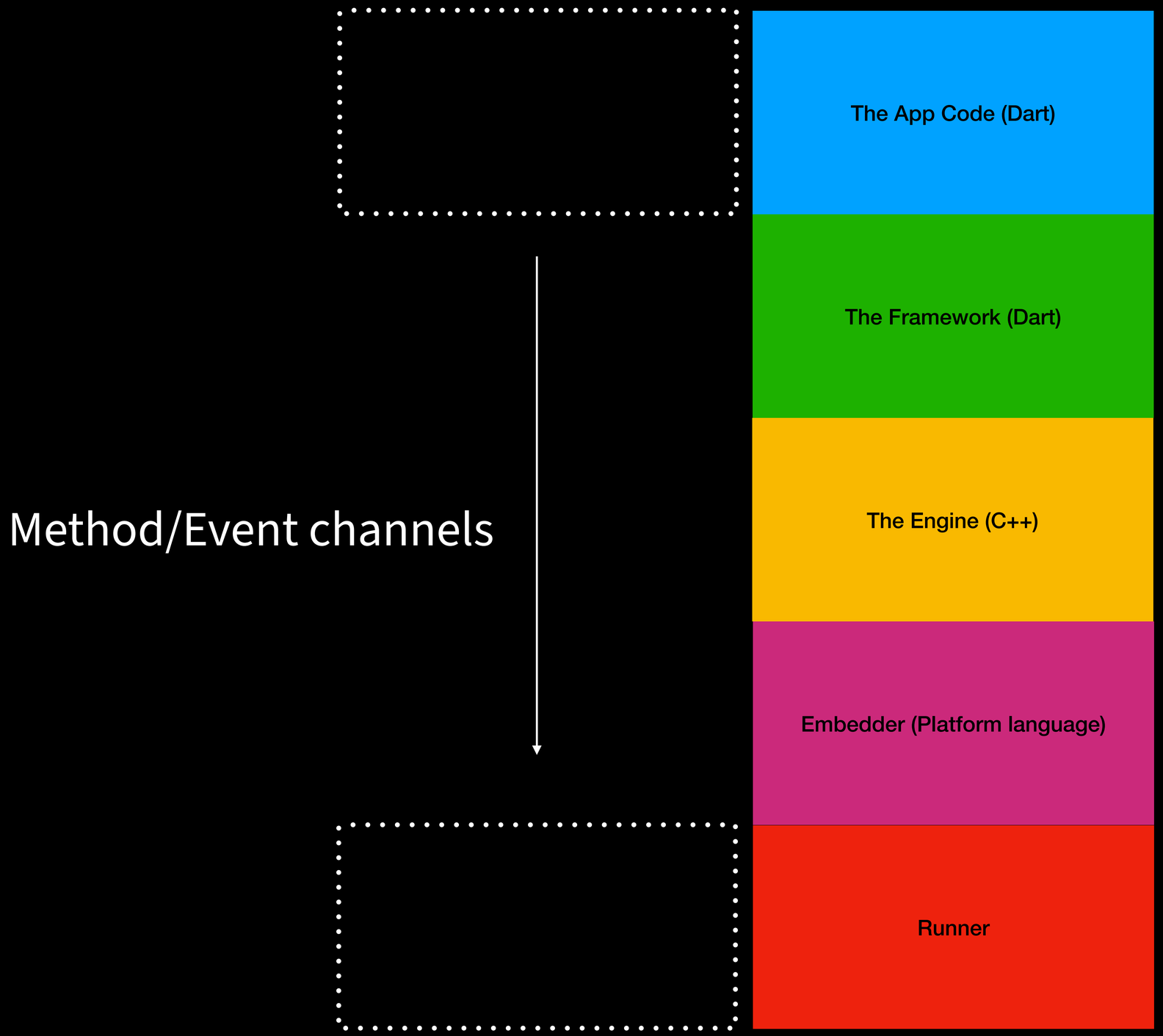
Credits to the Flutter team: Anatomy of an app (link)


Embedder (Platform language)
Runner

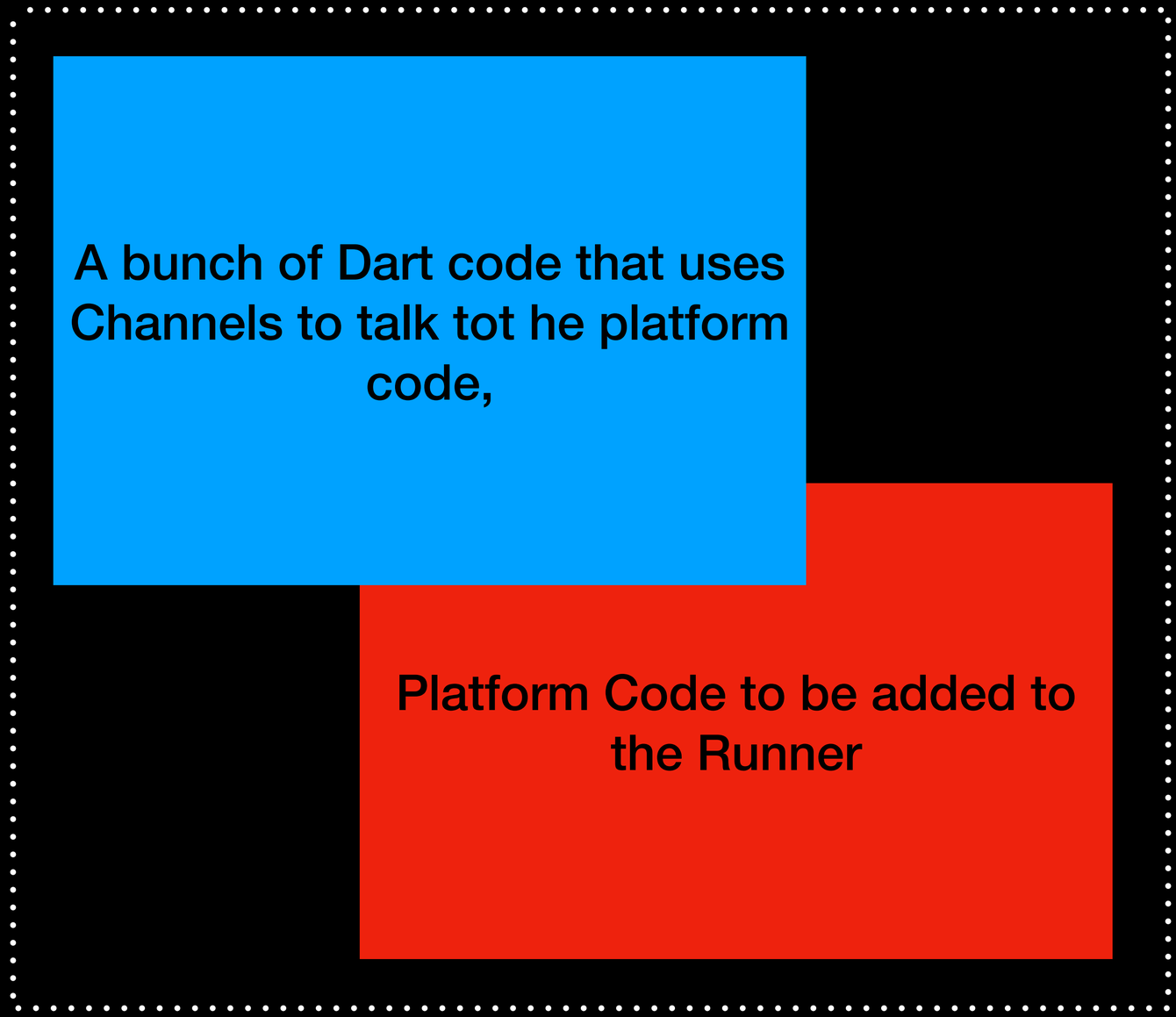
Lives in the Apps repository

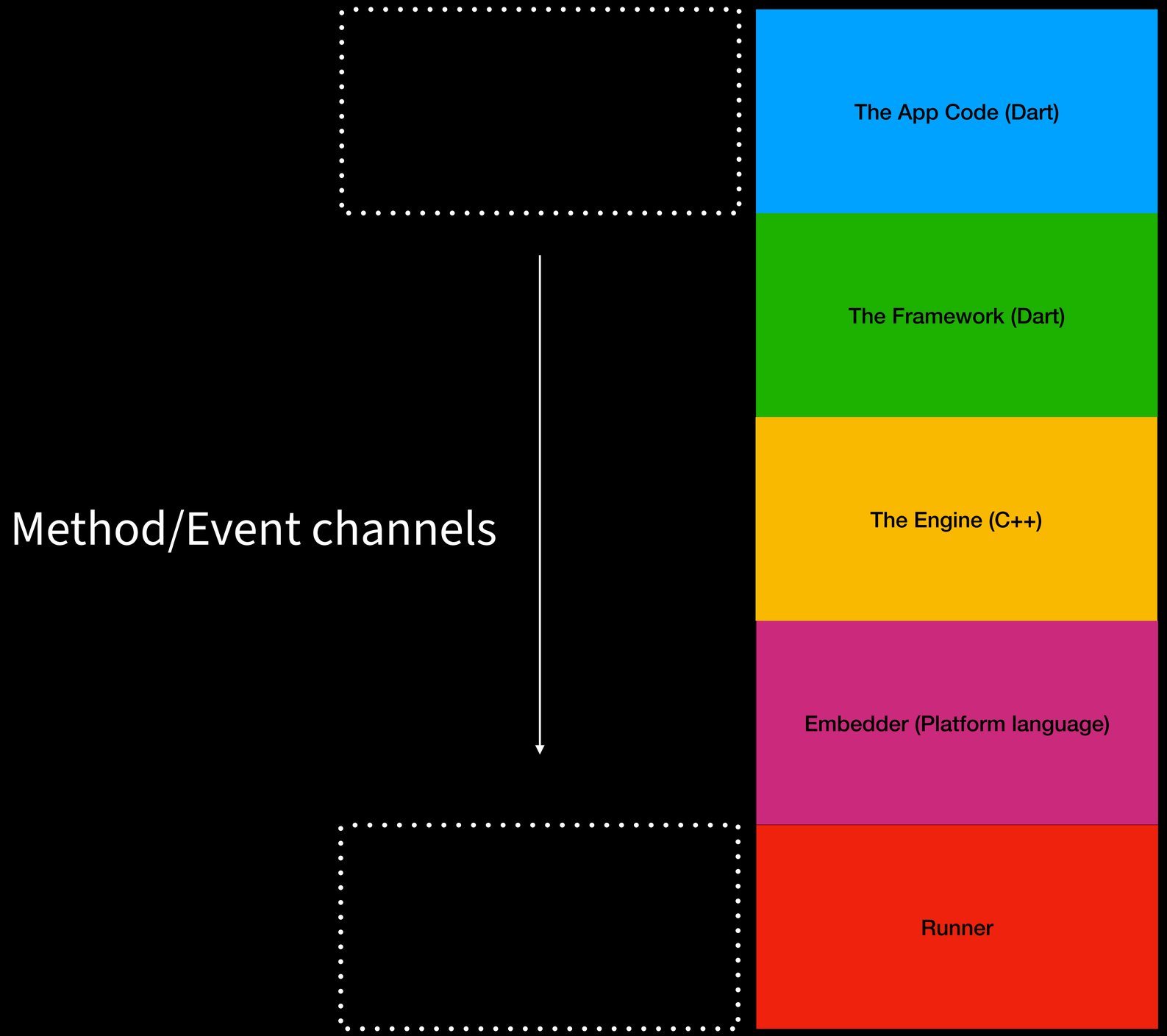


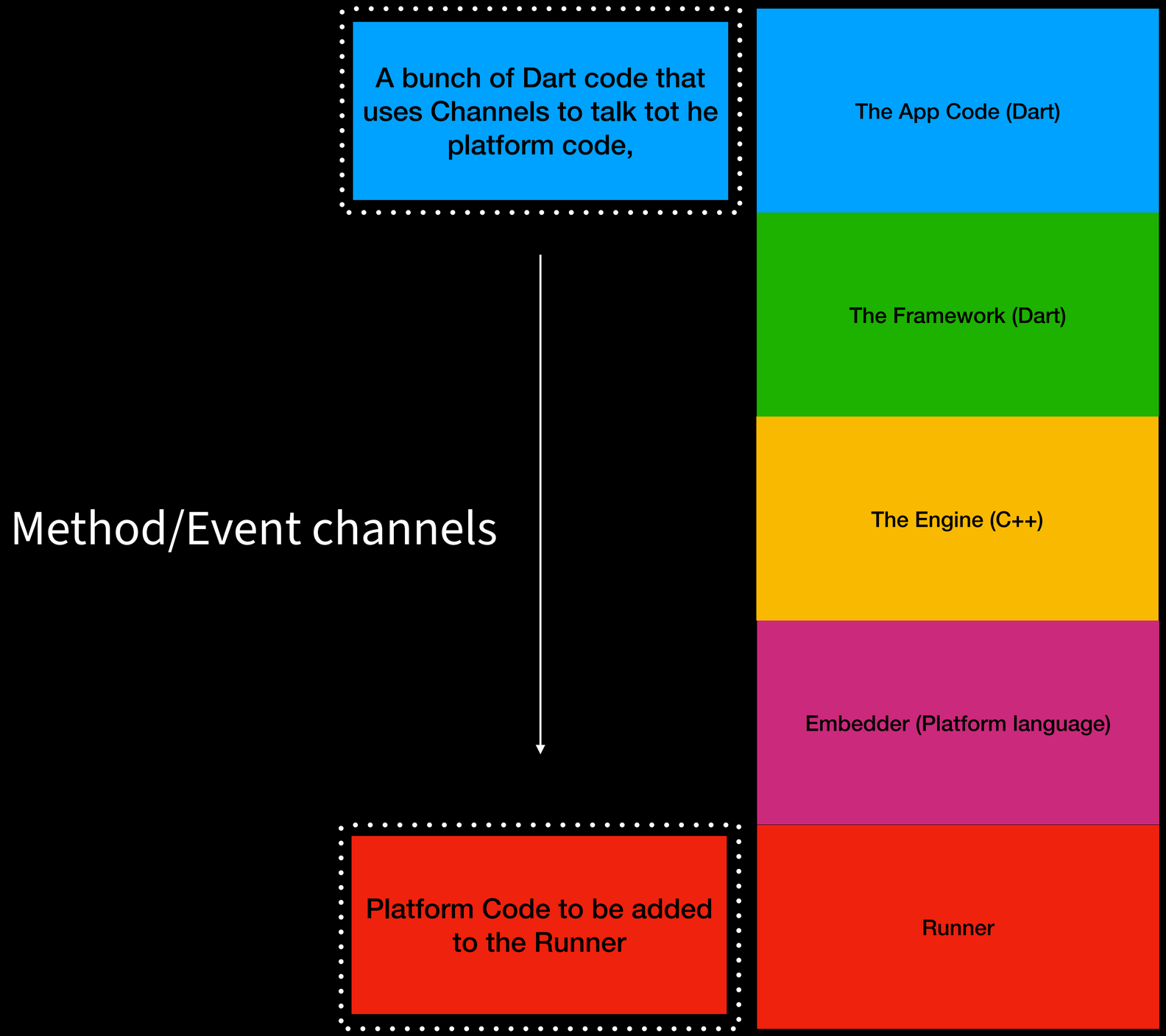

The App Code (Dart)
The Framework (Dart)
The Engine (C++)
Embedder (Platform language)
Runner


The App Code (Dart)
The Framework (Dart)
The Engine (C++)
Embedder (Platform language)
Runner
Method/Event channels

My Plugin

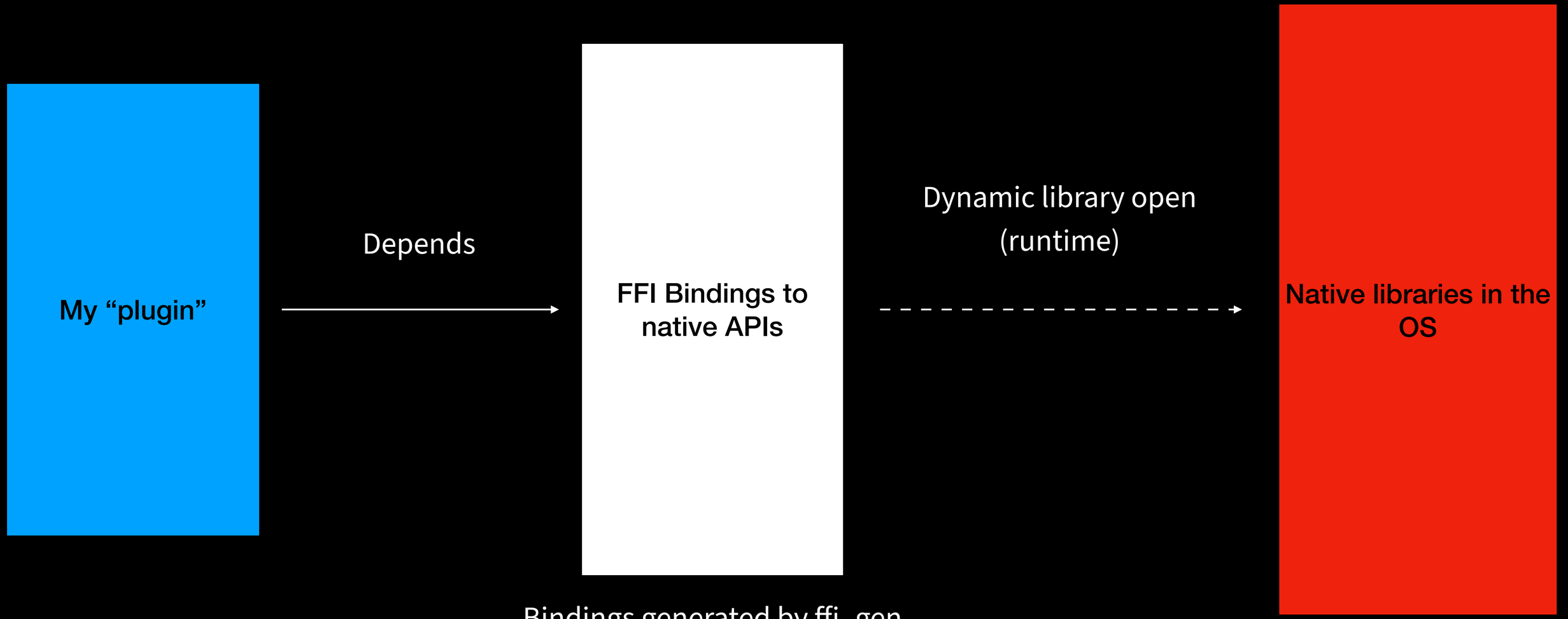

A bunch of Dart code that uses Channels to talk tot he platform code,
Platform Code to be added to the Runner

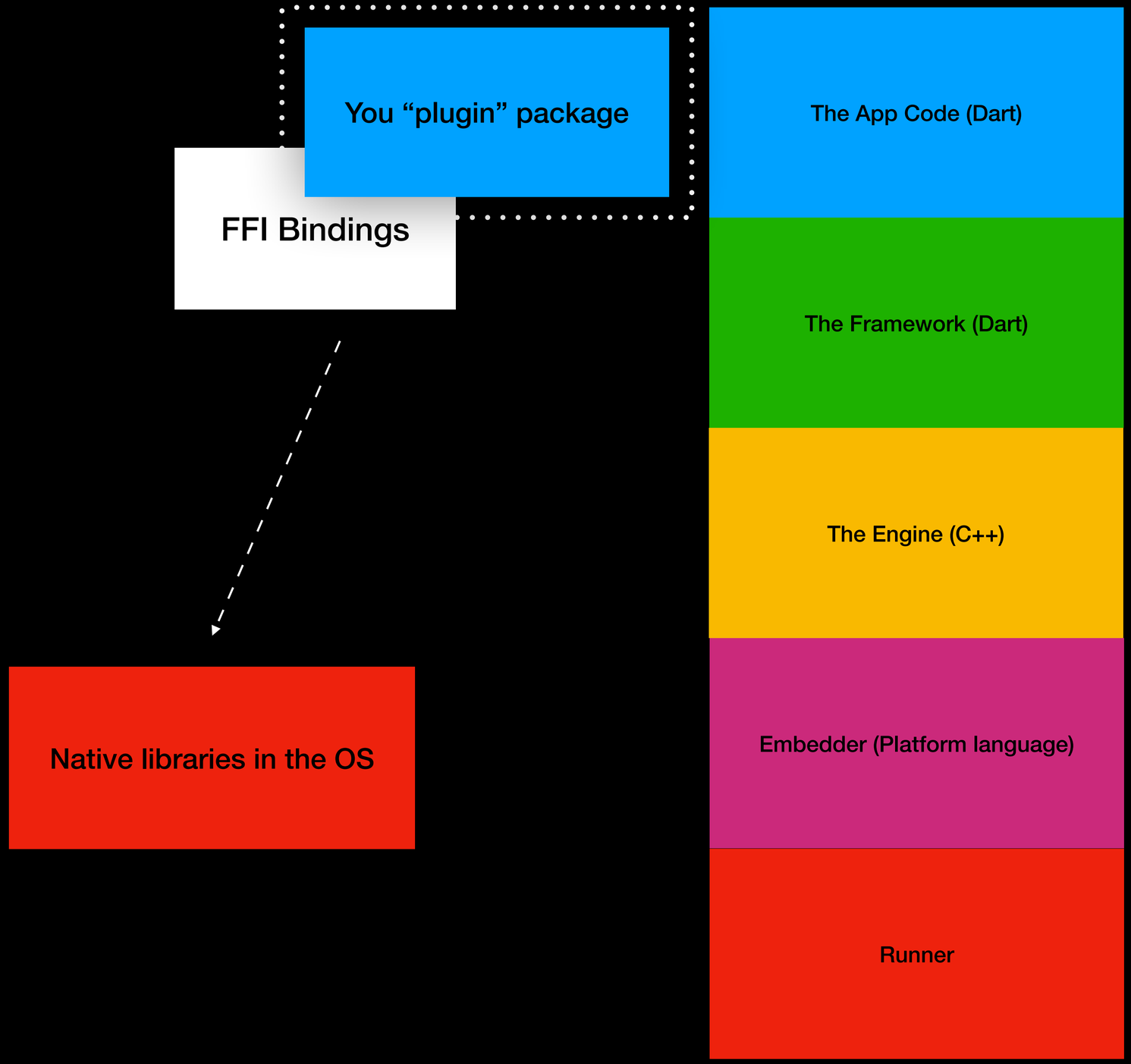

The App Code (Dart)
The Framework (Dart)
The Engine (C++)
Embedder (Platform language)
Runner
Method/Event channels


A bunch of Dart code that uses Channels to talk tot he platform code,
Method/Event channels
Platform Code to be added to the Runner
The App Code (Dart)
The Framework (Dart)
The Engine (C++)
Embedder (Platform language)
Runner

Part 2

Now lets do Dart-only “plugins”

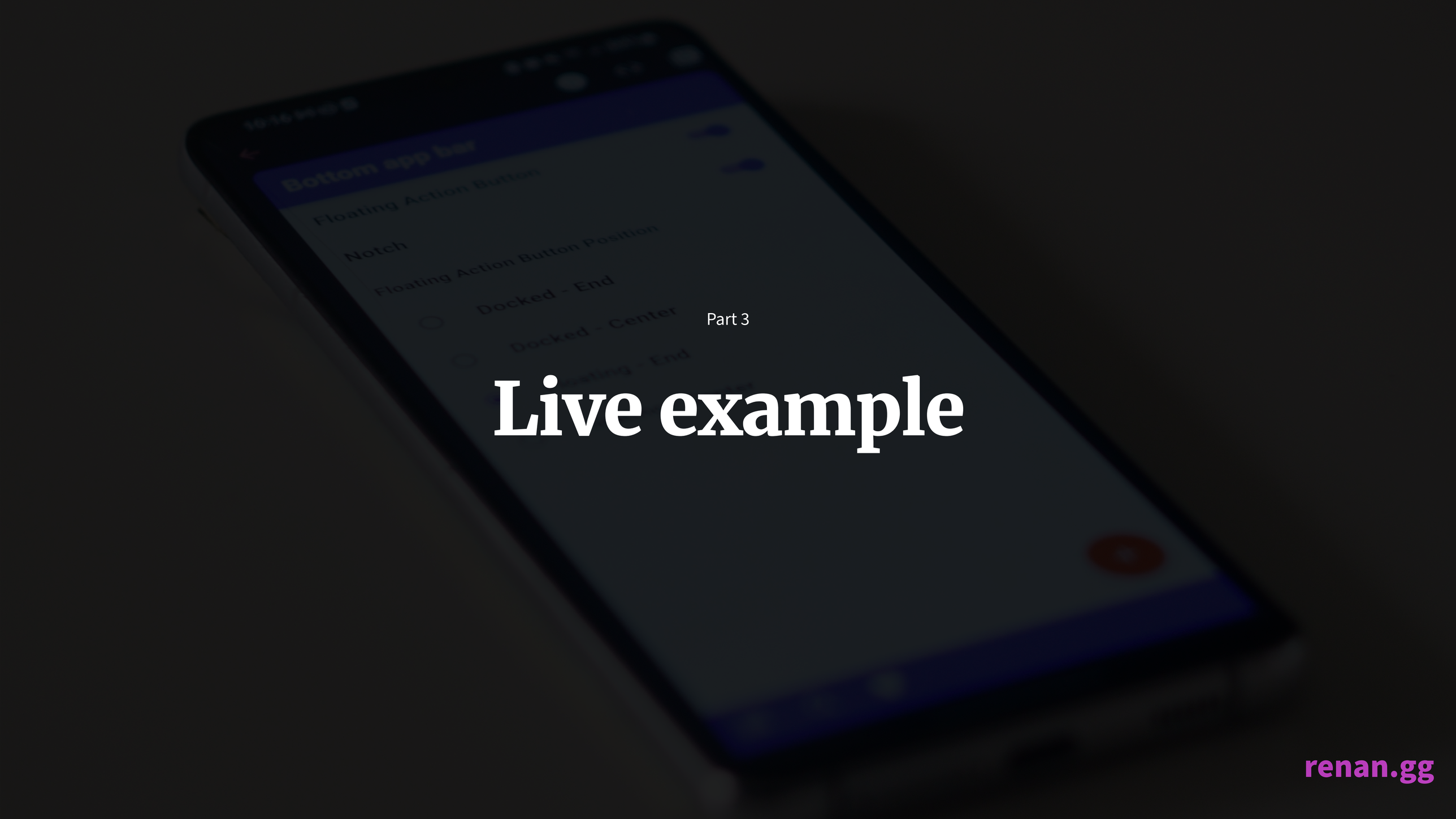

My “plugin”
Depends
FFI Bindings to native APIs
Dynamic library open (runtime)
Native libraries in the OS
Bindings generated by ffi_gen or packages like win32
Thanks Tim!


You “plugin” package
FFI Bindings
Native libraries in the OS
The App Code (Dart)
The Framework (Dart)
The Engine (C++)
Embedder (Platform language)
Runner

What changes?

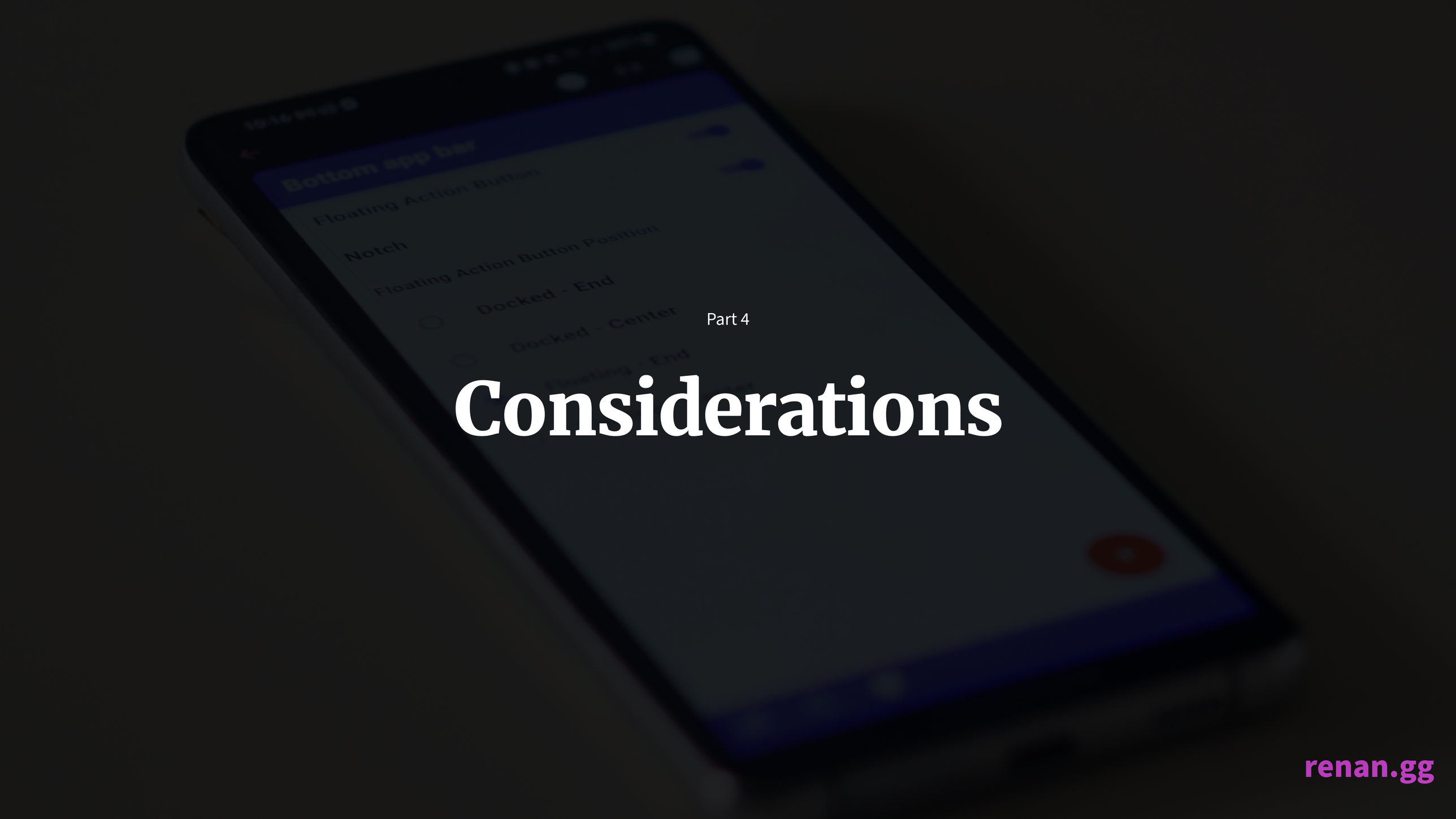
Differences to channels

- Calls are Type safe and Sync
- It is a Dart API, so no dependency on Flutter (in theory)
- Packages that exclusively call System Dylibs are not required to carry any native code



Part 3

Live example



Part 4

Considerations

Limitations

- Severe multithreading limitations
- It is unclear if the Apple Store will approve iOS apps that loads dynamic libraries via Dart

Continue learning

Objective-C and Swift interop using package:ffigen (link)

Rethinking Dart interoperability with Android (link)

Win32 package (link)

Example Repo (link)

Thank you

@reNotANumber

renan.gg