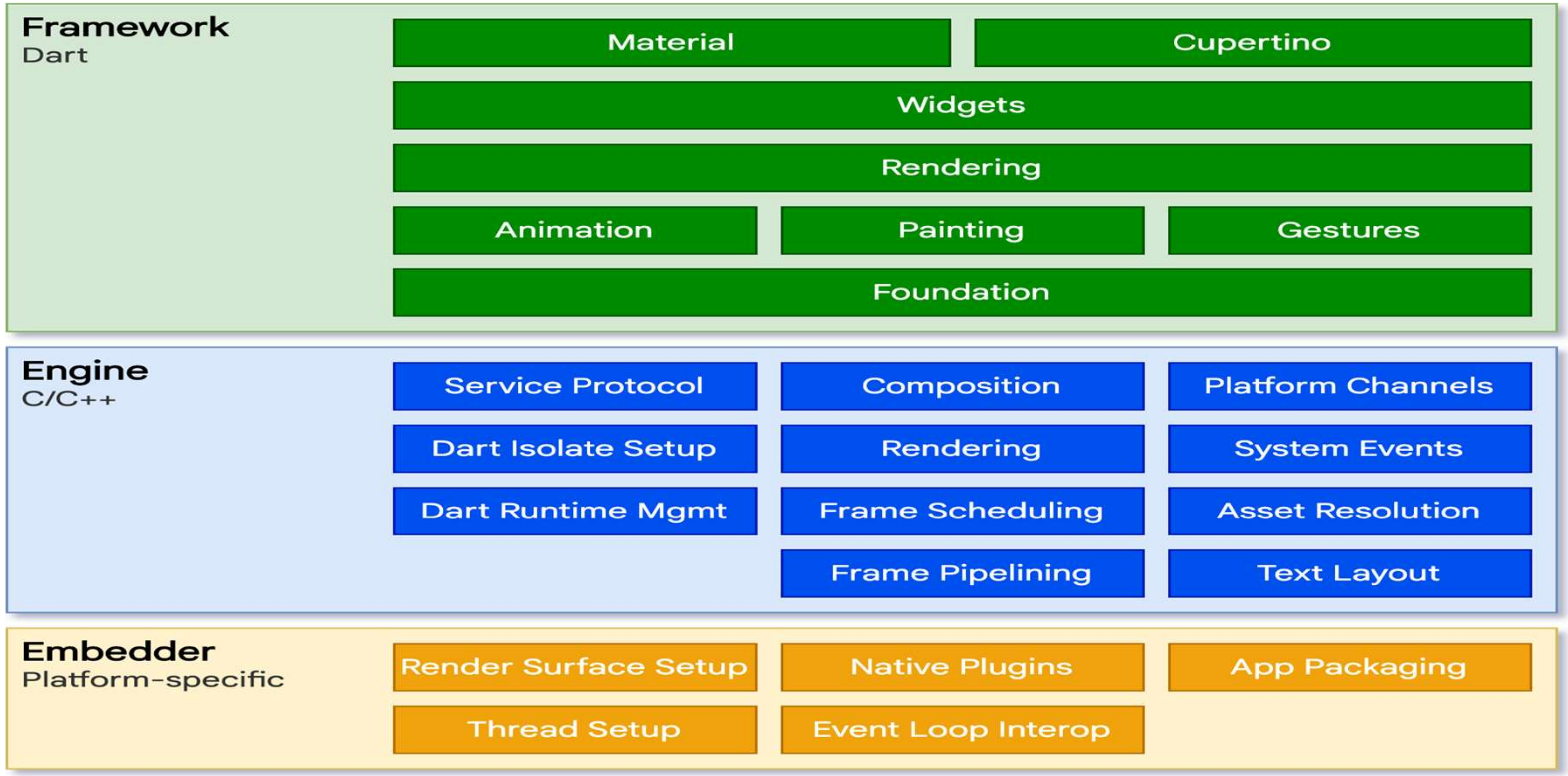
Flutter's architecture

- Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.



Embedder

- A platform-specific embedder provides an entrypoint; coordinates with the underlying operating system for access to services like rendering surfaces, accessibility, and input; and manages the message event loop.

- The embedder is written in a language that is appropriate for the platform: currently:-

   1. Java and C++ for Android,
   2. Objective-C/Objective-C++ for iOS and macOS,
   3. C++ for Windows and Linux.

- Using the embedder, Flutter code can be integrated into an existing application as a module, or the code may be the entire content of the application. Flutter includes a number of embedders for common target platforms, but other embedders also exist.

Engine

At the core of Flutter is the Flutter engine, which is mostly written in C++ and supports he primitives necessary to support all Flutter applications. The engine is responsible for rasterizing composited scenes whenever a new frame needs to be painted.

It provides the low-level implementation of Flutter’s core API, including graphics (through Skia), text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain.

Skia

Skia is an open source 2D graphics library which provides common APIs that work across a variety of hardware and software platforms. It serves as the graphics engine for Google Chrome and Chrome OS, Android, Flutter, and many other products.

But Why Skia is So Important in Flutter?

Flutter's approach is different from most other cross-platform frameworks. While React Native, Xamarin, and Titanium try to bring out native (Android/ iOS) UI components, Flutter decided to render UI components by itself. That means when you see a UI component—a button for example, it's not an Android or iOS button, it's a Flutter button rendered directly by the Skia framework.

- **Platforms**
  - **Windows 7, 8, 8.1, 10**
  - **macOS 10.10.5 or later**
  - **iOS 11 or later**
  - **Android 4.1 (JellyBean) or later**
  - **Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, or Fedora Linux 32+**

Flutter framework,

- Typically, developers interact with Flutter through the **Flutter framework**, which provides a modern, reactive framework written in the Dart language. It includes a rich set of platform, layout, and foundational libraries, composed of a series of layers. Working from the bottom to the top, we have:

Basic foundational classes, and building block services such as animation, painting, and gestures that offer commonly used abstractions over the underlying foundation.

The rendering layer provides an abstraction for dealing with layout. With this layer, you can build a tree of renderable objects. You can manipulate these objects dynamically, with the tree automatically updating the layout to reflect your changes.

The widgets layer is a composition abstraction. Each render object in the rendering layer has a corresponding class in the widgets layer. In addition, the widgets layer allows you to define combinations of classes that you can reuse. This is the layer at which the reactive programming model is introduced.

The Material and Cupertino libraries offer comprehensive sets of controls that use the widget layer’s composition primitives to implement the Material or iOS design languages.