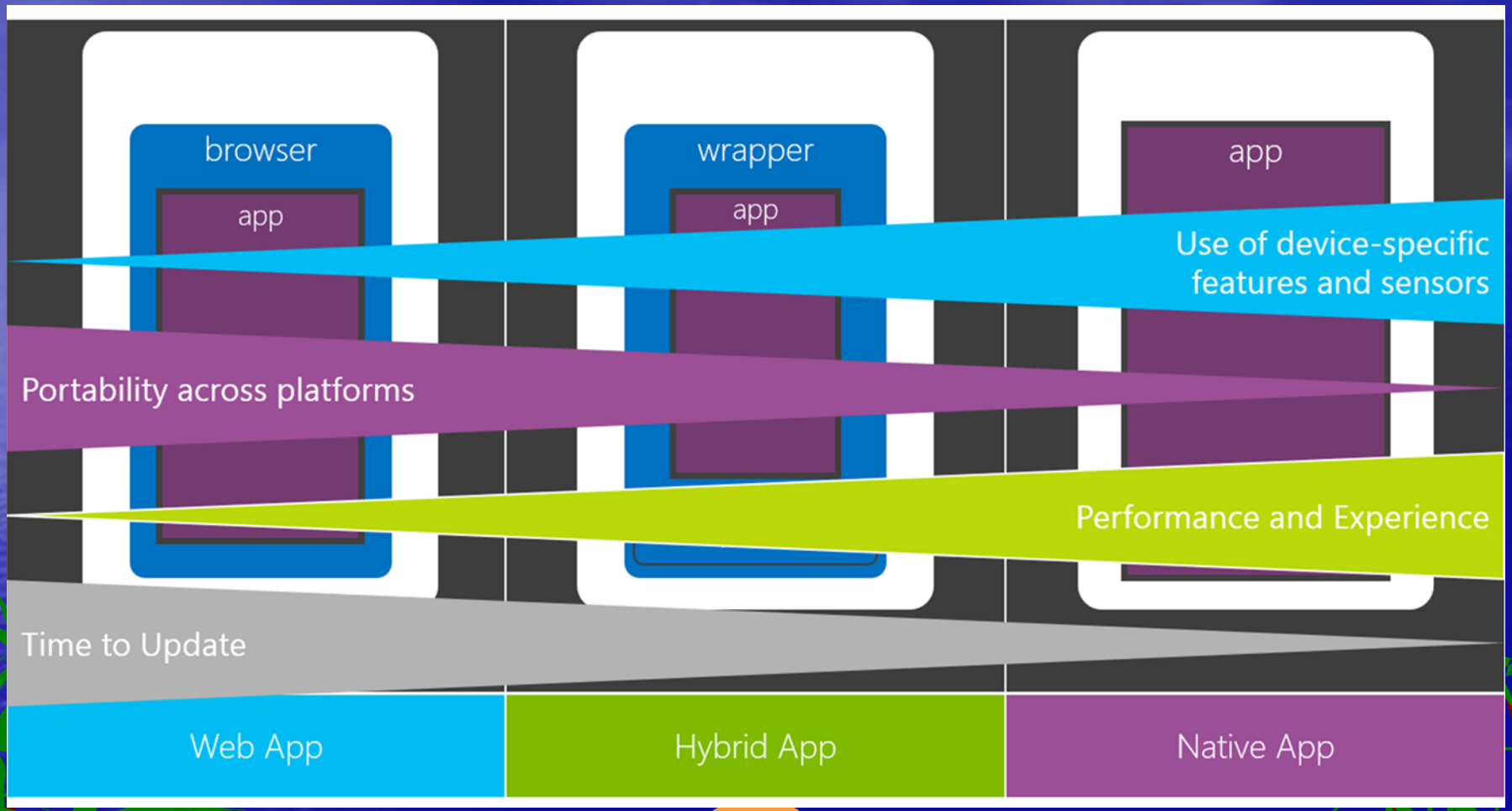
Mobile Computing

The Flutter Framework
Stateless Widgets

What is Flutter

- Flutter is an open-source application development kit created by Google
  - Ultimately it aims applications for
    - Mobile (Android and iOS)
    - Web (deployed in servers, embedded in browsers, PWAs, SPAs)
    - Desktop (still in beta)
      - Windows (needs VS 2019)
      - MacOS (needs XCode)
      - Linux (needs CLang, CMake, GTK)
    - The experimental Google OS Fuchsia
  - It uses the Dart programming language and runtime
  - Tools, Resources and Installation from
    - https://flutter.dev

Mobile Application Approaches



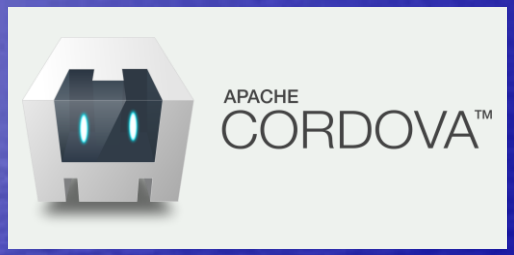
Web Applications for Mobile

- Same technologies as other web applications
  - Run on the device browser and a remote server

  - Limited in some features
    - Use of device hardware and controls
    - UX different from native
    - Performance

Popular Hybrid Frameworks

- Hybrid Apps run on the device and off-line
  - More integrated but still with some drawbacks
    - Non-native experience
    - Performance problems concerning certain components

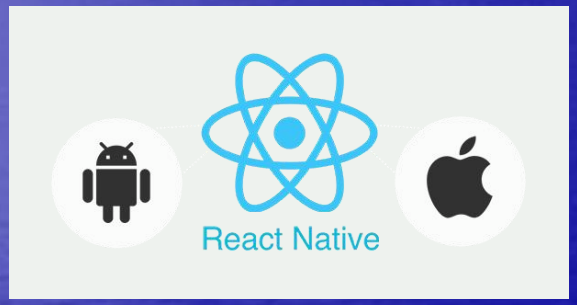


+

RhoMobile
(Tau technologies)

Near Native (Hybrid-native) technologies

- Produce and execute on native (or VM) code
  - UI near the native, facilities to access the original API
    - With good performance



JSX, JS
. web like separated UI specification

C#
. architecture pattern oriented (MVVM)
. separated UI specification
. Xamarin.Forms rendered with Android / iOS native views

Dart
. widgets

Flutter Development Tools

- You can install your development tools
  - Windows, MacOS, or Linux
    - Flutter SDK, Dart SDK, Android and/or iOS SDK
    - CLI tools
  - With plugins, high level IDEs


Android Studio

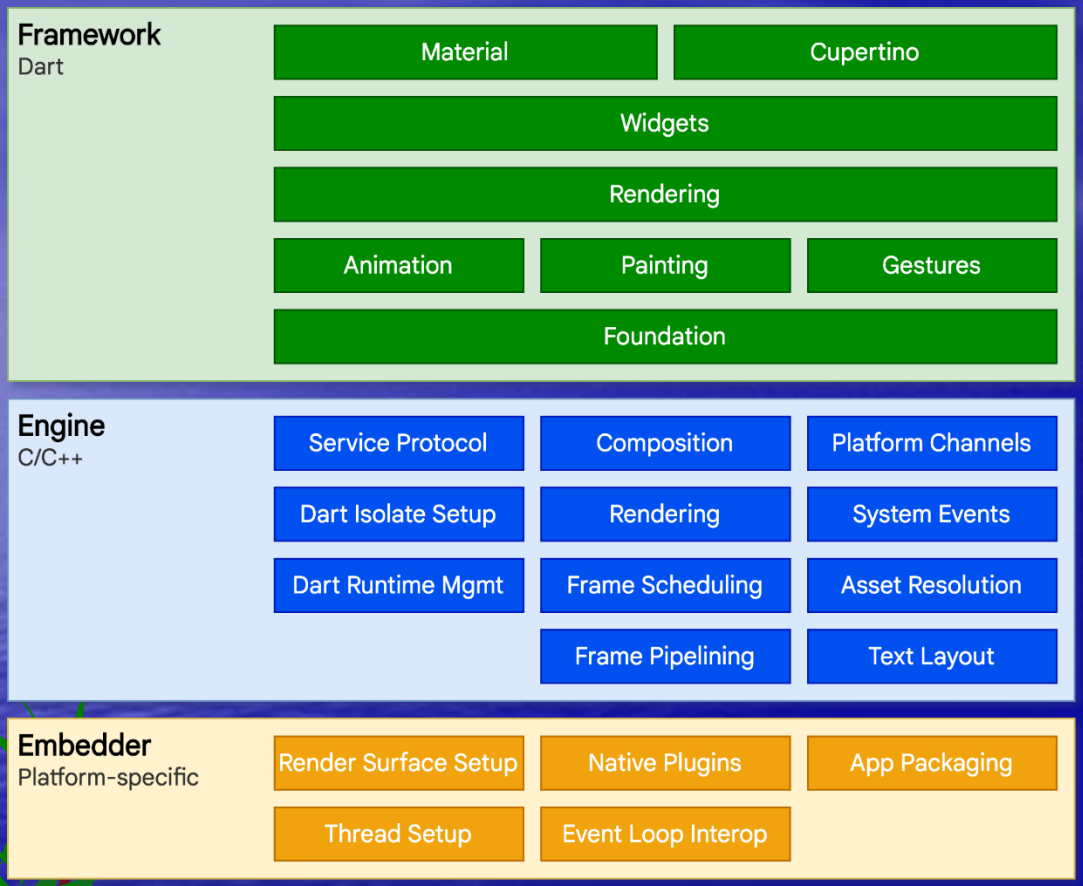

IntelliJ Idea


Visual Studio Code


Emacs

  - Android app can be tested and executed in any OS with the Android SDK
  - iOS app can only be completed and executed through a Mac with XCode and the iOS SDK
    - A paid development license from Apple is needed (for devices)

Flutter Layered Architecture



Flutter Framework – Developer interface
Main app code uses the 2 top layers (everything is a widget)
Customizations can involve classes from the previous-to-last layer (animations, input gestures, drawing, …)

Flutter Engine – Set of primitives used by the Framework. It comprises graphics (Skia), text rendering, file and network I/O, plugin architecture, and the Dart run-time.
This layer is exposed in the Framework as the low-level dart:ui package

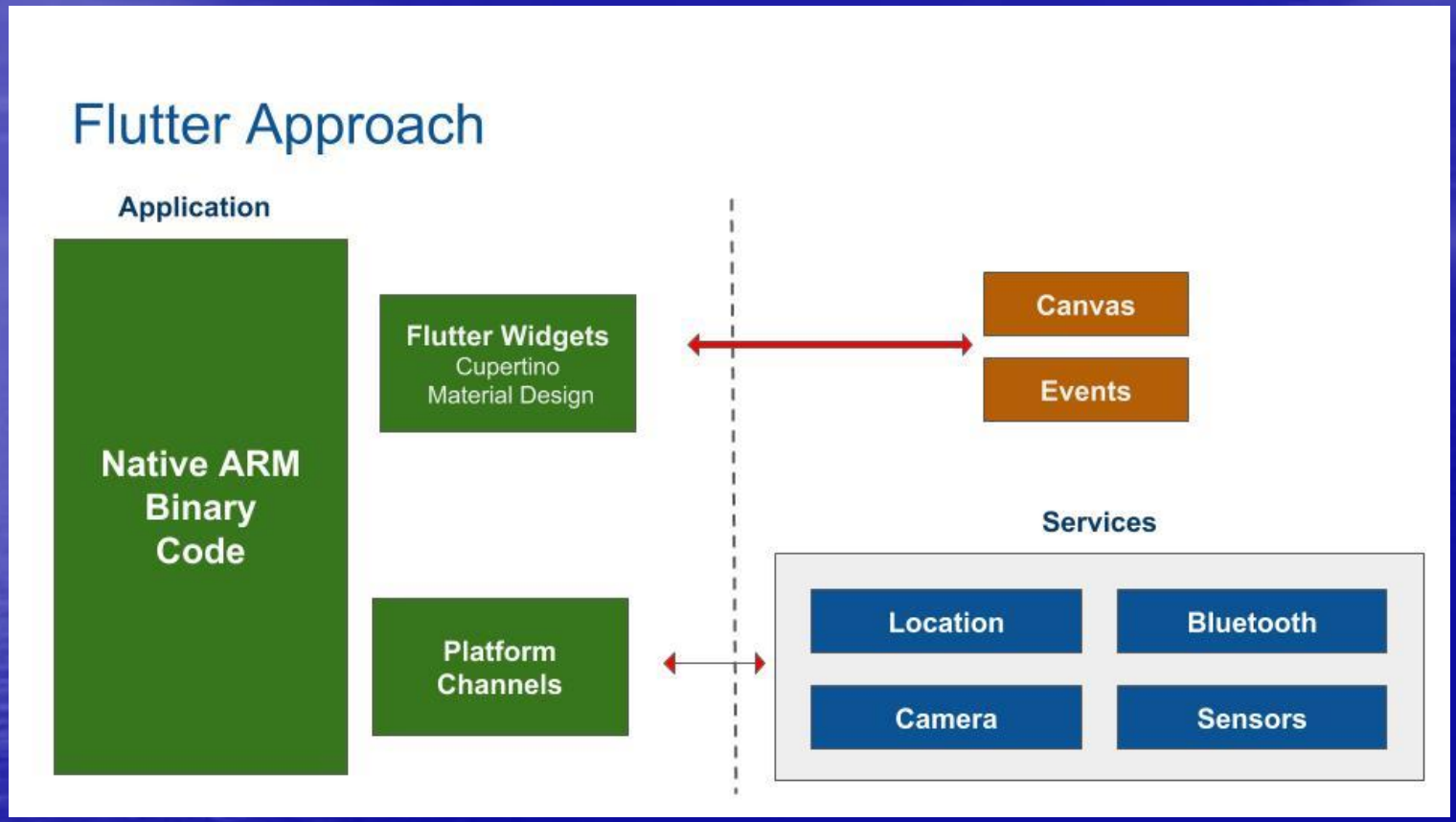
Flutter Embedder – code layer that interfaces the native OS and their services. It provides a connection with the native message loop, allowing the flutter app to run on the top of a native app.
It is written in Java (Android) or Objective-C (iOS), and C++ (all OS's)

Flutter follows a reactive model of UI. The UI is a tree of widgets, with a separated associated state. A change in state automatically triggers a UI update.

$$UI = f(state)$$

(https://flutter.dev/docs/resources/architectural-overview)

Flutter and Native OS



Flutter UI through Widgets – draw direct in the screen represented by a canvas receive the events generated by the user interaction

Other functionalities need calling and data exchange with native services that is done using Flutter platform channels

Flutter App and Widgets

- Flutter apps start from the Dart main() function
  - A call to the Framework runApp(…) should be made
    - The parameter should be a widget derived object, with the UI of the app's home page

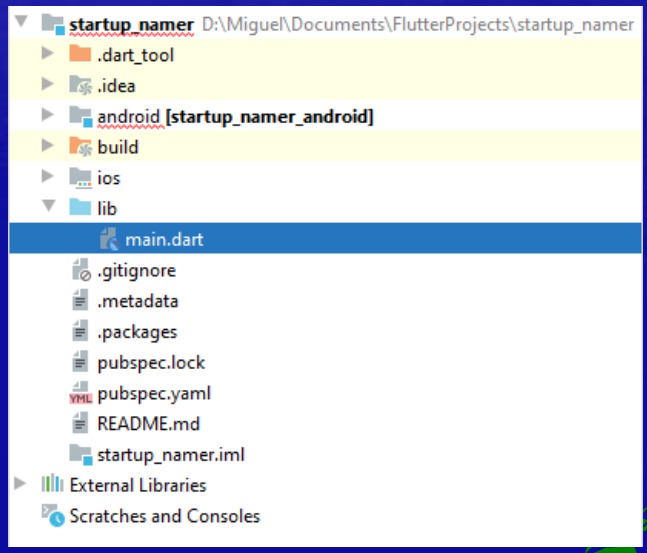
```
import 'package:flutter/… ';    // needed Dart and Flutter imports

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp (
      …
    );
  }
}
```

one of the Framework top level widgets:
WidgetApp
CupertinoApp
MaterialApp

(https://dart.dev/guides/language/language-tour)

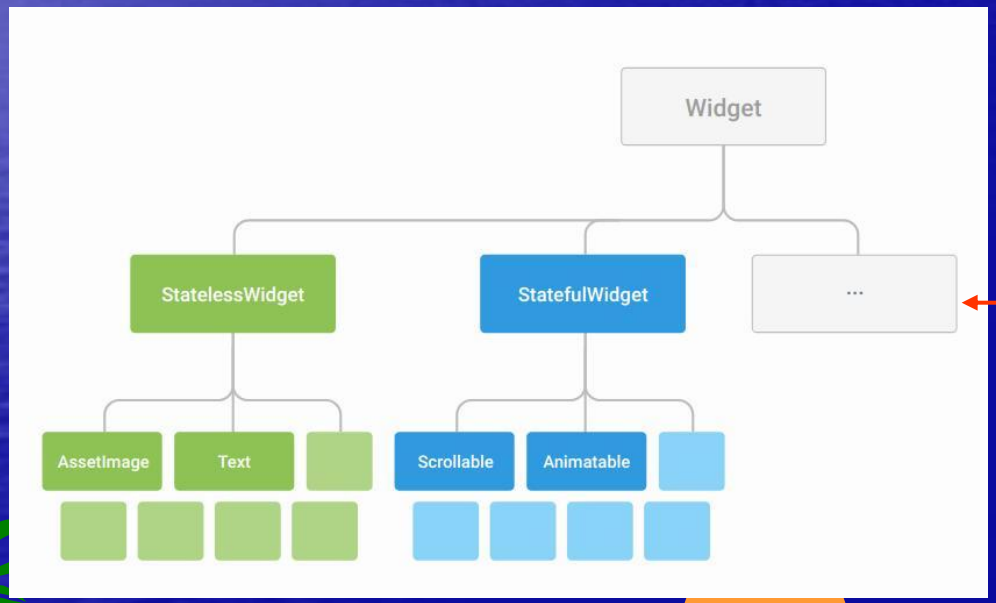


Flutter project structure

- The top widget is usually one of the App widgets of the Framework

Widgets

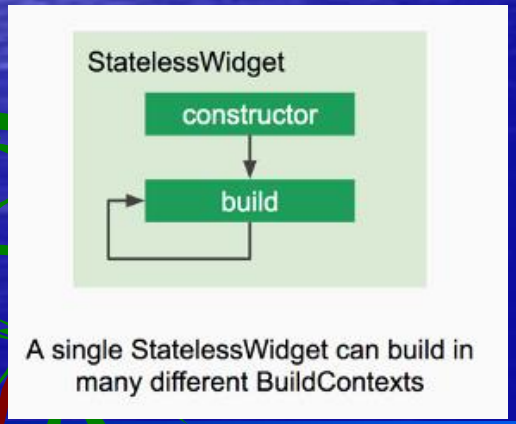
- All the UI is made by a tree of Widgets
  - Not only displayed objects, but also almost anything related to a UI is a Widget (e.g., the GestureDetector or the Align widget)
    - Almost all widgets are a StatelessWidget or a StatefulWidget



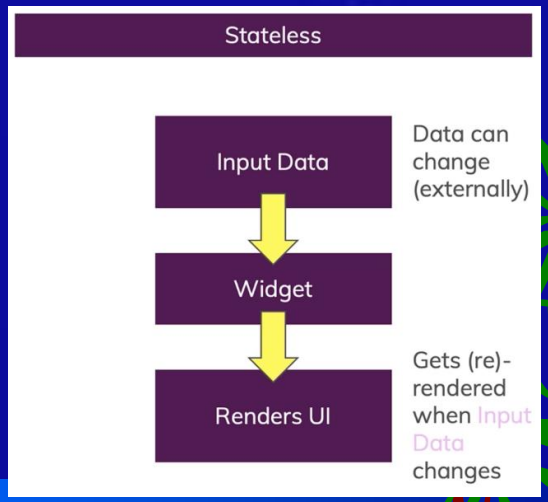
we have also in the Framework:
. PreferredSizeWidget
. RenderObjectWidget
. ProxyWidget

Stateless Widgets

- **Stateless widgets are immutable**
  - They receive configuration data through the constructor
    - The constructor calls the build(…) method that returns the widget object
    - The build(…) cannot be called again
  - To redraw a StatelessWidget a new instance must be created

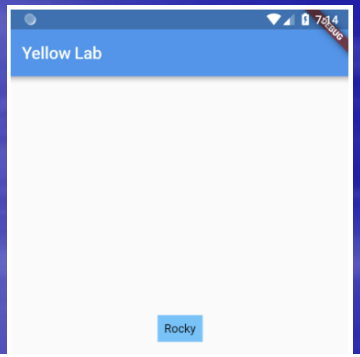


Widgets receive a BuildContext in the build( ) method
It is a reference to the location of a Widget in the UI tree
It can contain properties concerning the widgets rendering



Stateless Example

One dog

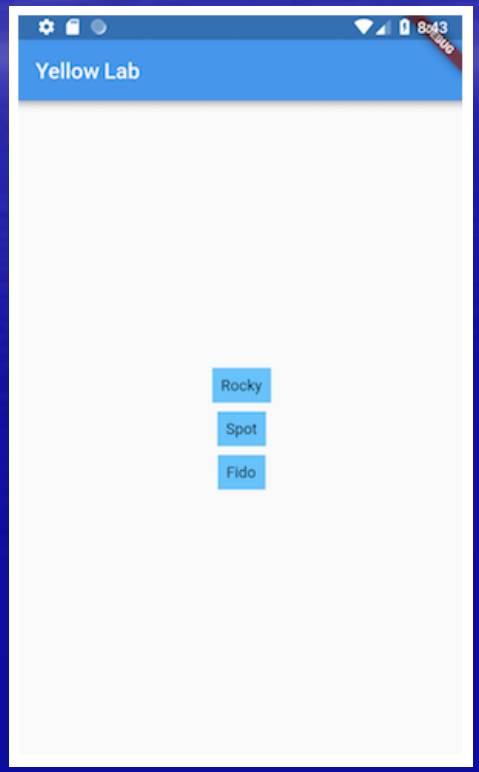



```dart
import 'package:flutter/material.dart';

void main() {
 runApp(new DogApp());
}

class DogApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'My Dog App',
     home: Scaffold(
       appBar: AppBar(
         title: Text('Yellow Lab'),
       ),
       body: Center(
         child: DecoratedBox(
           decoration: BoxDecoration(color: Colors.lightBlueAccent),
           child: Padding(
             padding: const EdgeInsets.all(8.0)
             child: Text('Rocky'),
           ),
         ),
       ),
     ),
   );
 }
```

Three Dogs in a column




```dart
...
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            DogName('Rocky'),
            SizedBox(height: 8.0),
            DogName('Spot'),
            SizedBox(height: 8.0),
            DogName('Fido'),
          ],
        ),
      ),
    );
  }
}
```

composition pattern

```dart
class DogName extends StatelessWidget {
 final String name;

const DogName(this.name);

 @override
 Widget build(BuildContext context) {
   return DecoratedBox(
     decoration: BoxDecoration(color: Colors.lightBlueAccent),
     child: Padding(
       padding: const EdgeInsets.all(8.0),
       child: Text(name),
     ),
   );
 }
}
```