

Flutter Architecture and State Management

Sung Pham 31 Aug, 2020

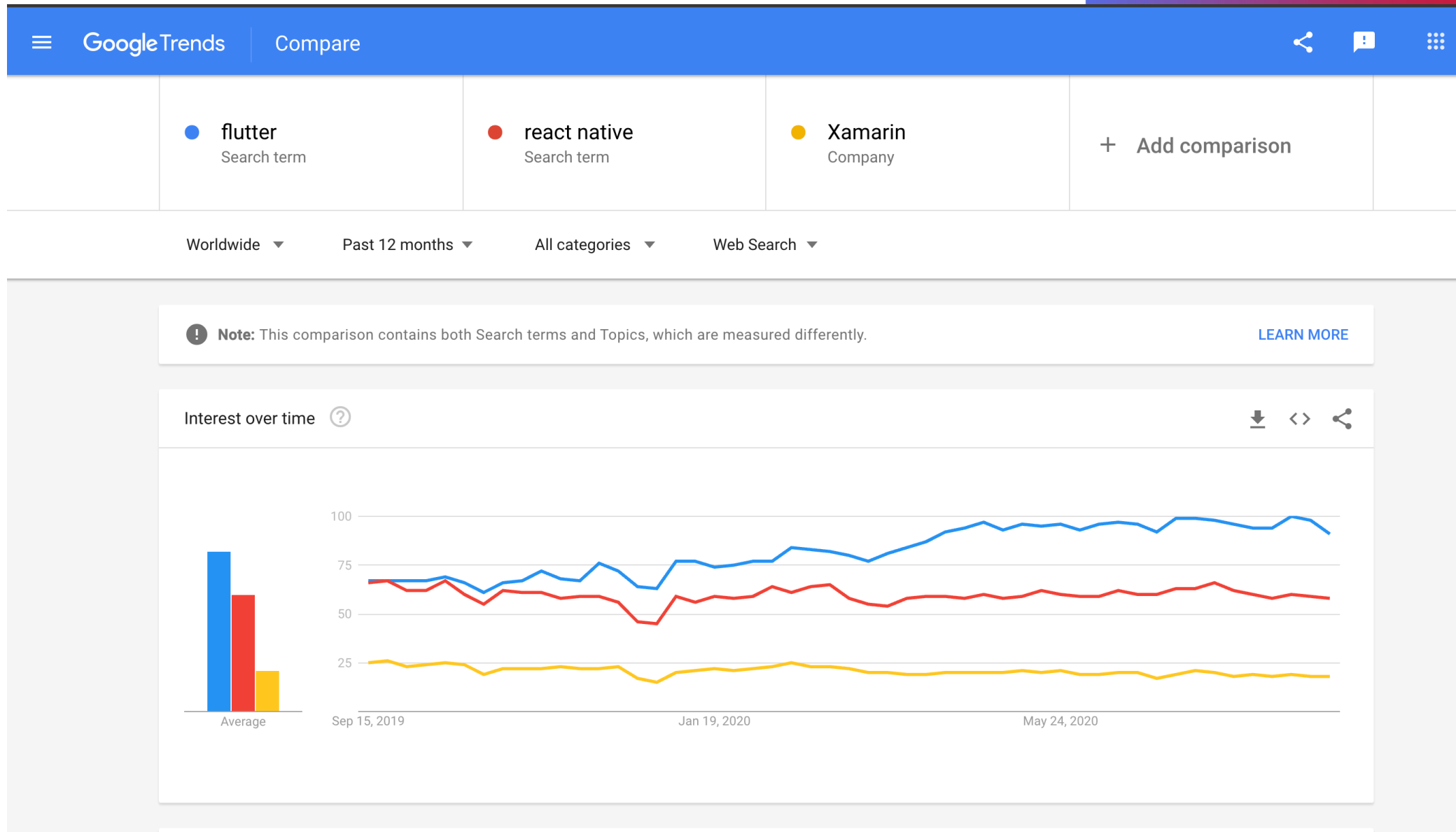


Outline

1. Why Flutter?
2. Flutter Application Architecture
3. State management solutions
4. Demo
5. Q&A




1. Why Flutter?



1. Why Flutter?

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

[Read the guide](#)


 flutter / flutter

Watch 3.1k Star 101k Fork 13.9k

Code Issues 5k+ Pull requests 179 Actions Projects 129 Wiki Security Insights

master 78 branches 376 tags

Go to file Add file Code

 tvolkert Small documentation update (#65392) 54ade88 2 hours ago 20,366 commits

.github Fix per [probot/no-response#25](#) (#65042) 6 days ago

bin Roll Engine from 3c8b9c4c52ee to d1d848e8421e (15 revisions) (#65042) 12 hours ago

About

Flutter makes it easy and fast to build beautiful apps for mobile and beyond.


[flutter.dev](#)

mobile android ios

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

[Read the guide](#)


 facebook / react-native

Watch 3.7k Star 90k Fork 19.9k

Code Issues 786 Pull requests 251 Actions Projects Wiki Security Insights

master 72 branches 347 tags

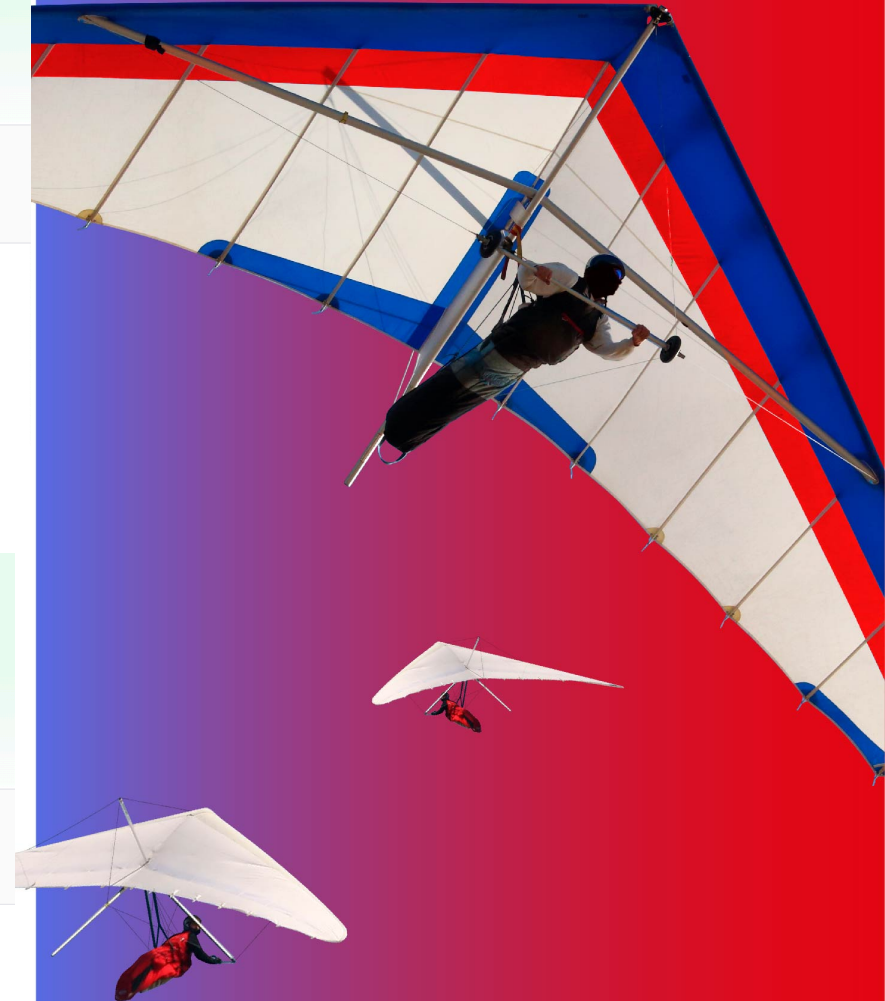
Go to file Add file Code

 shergin and facebook-github-bot Fabric: Better error reporting in UIView... 7e89934 8 hours ago 20,912 commits

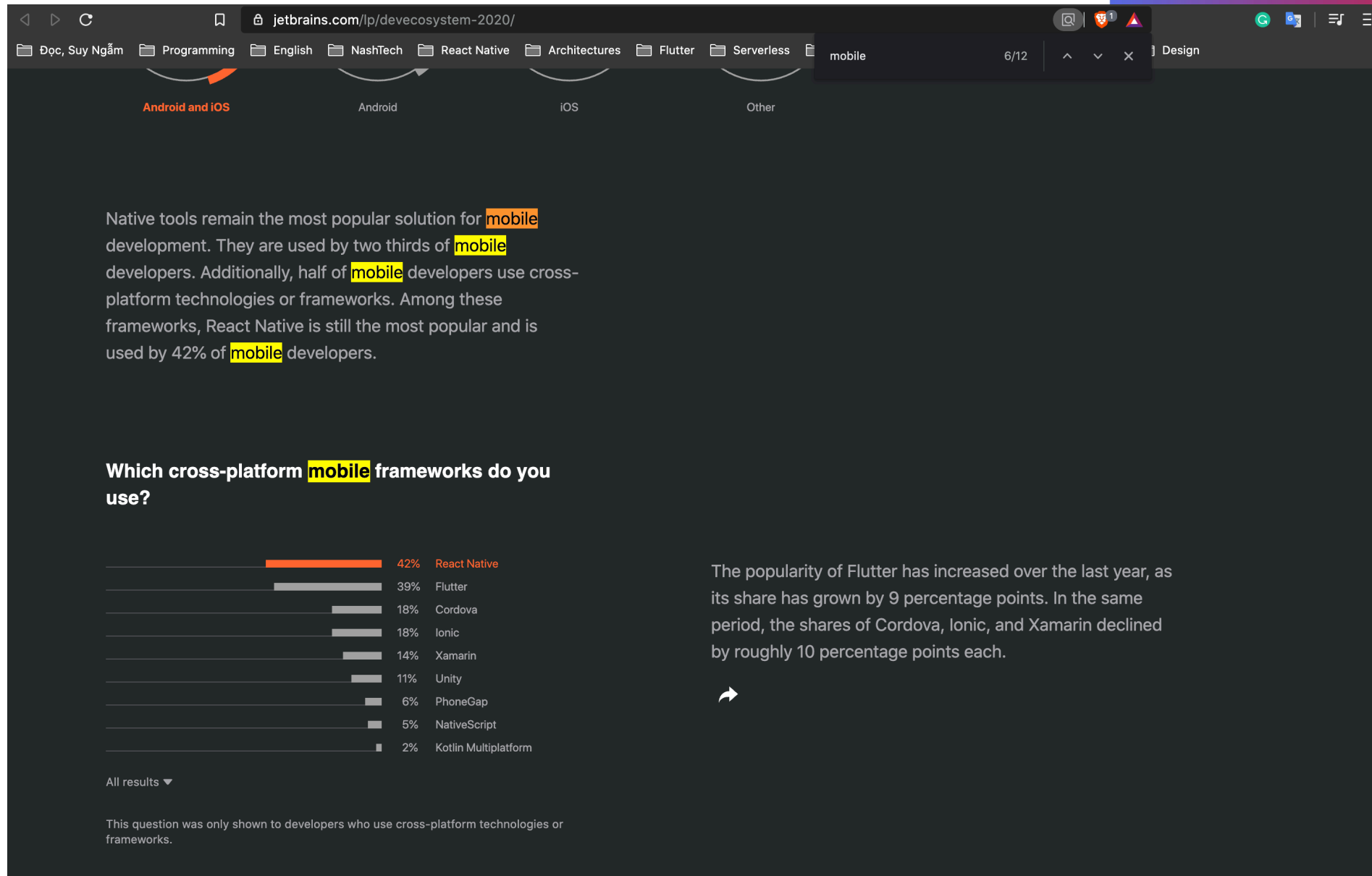
About

A framework for building native apps with React.

[reactnative.dev](#)



1. Why Flutter?



1. Why Flutter?

- ✓ **Flutter is backed by Google**

Its adoption among popular cross-platform mobile development tools

- ✓ **Speed and performance**

Fast, smooth, predictable

- ✓ **Flexibility**

Customization, control

- ✓ **Its own unique widgets**

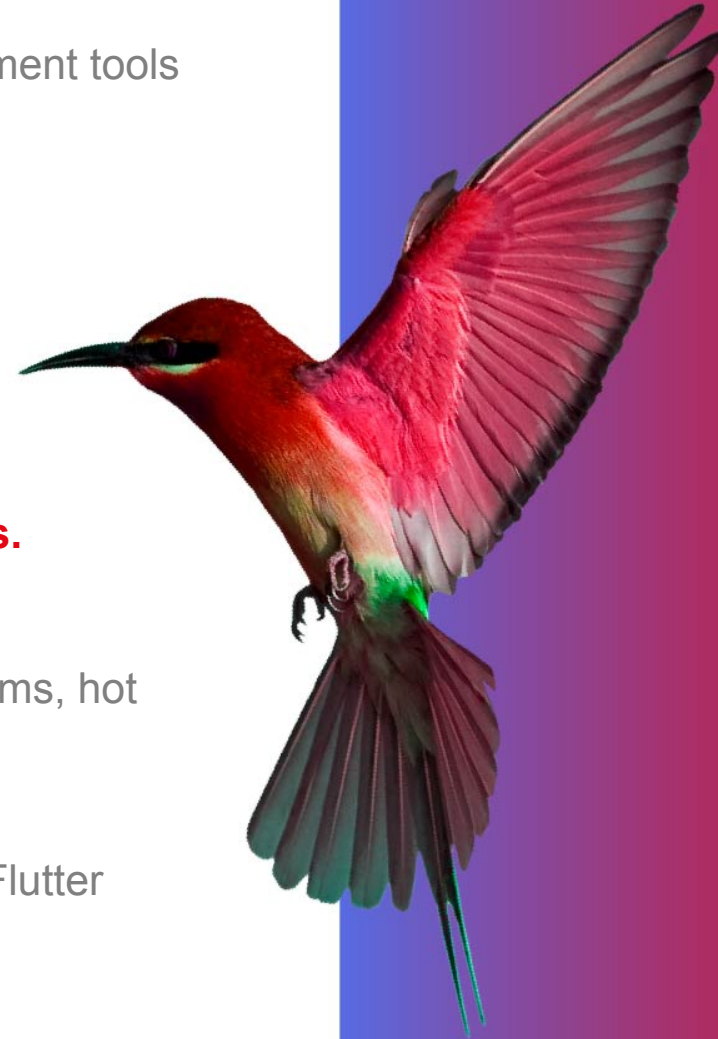
Without depending on platform-specific UI components.

- ✓ **Less time to market**

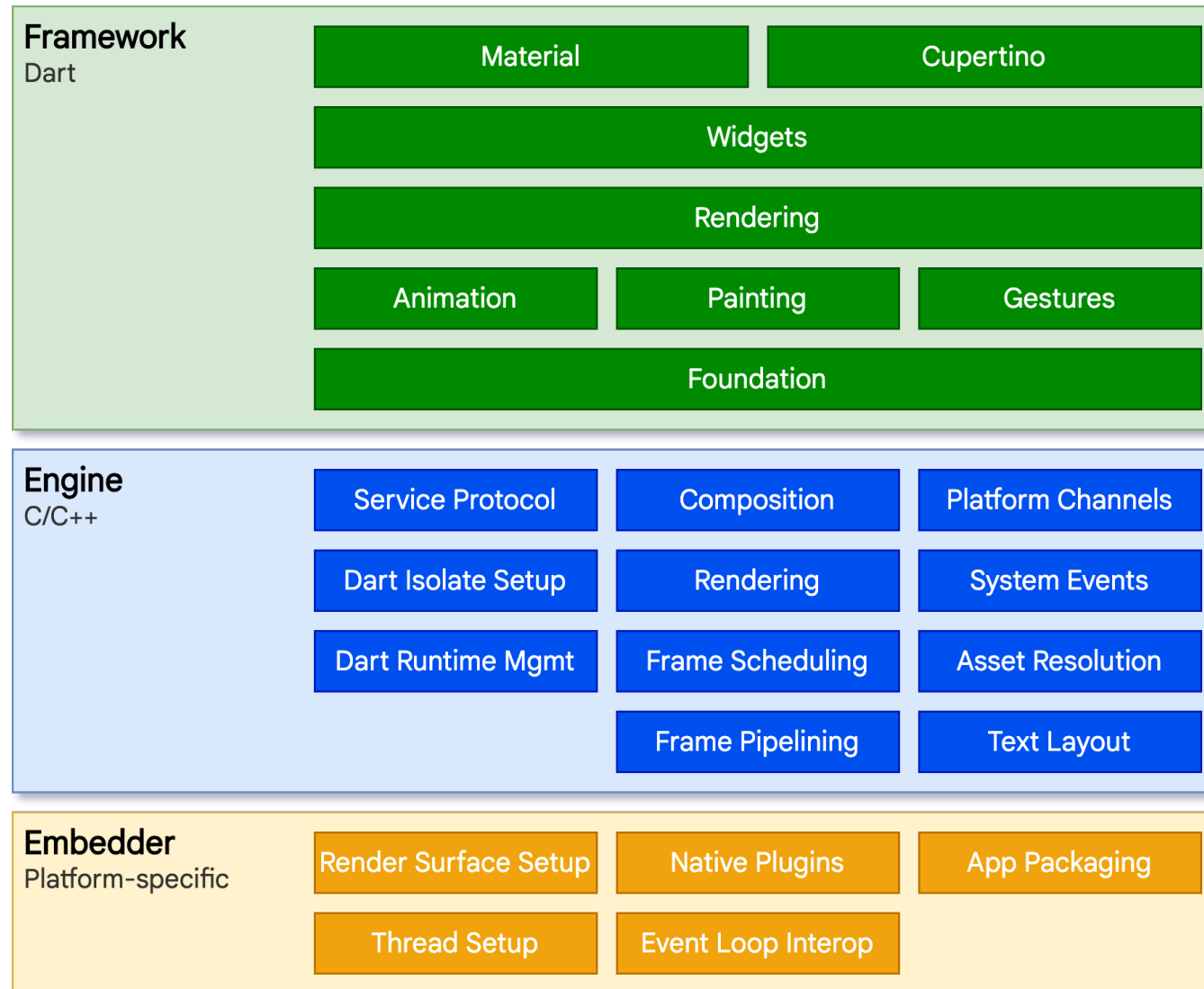
Rapid development, free, single sources for multiple platforms, hot reload...

- ✓ **Used by big companies**

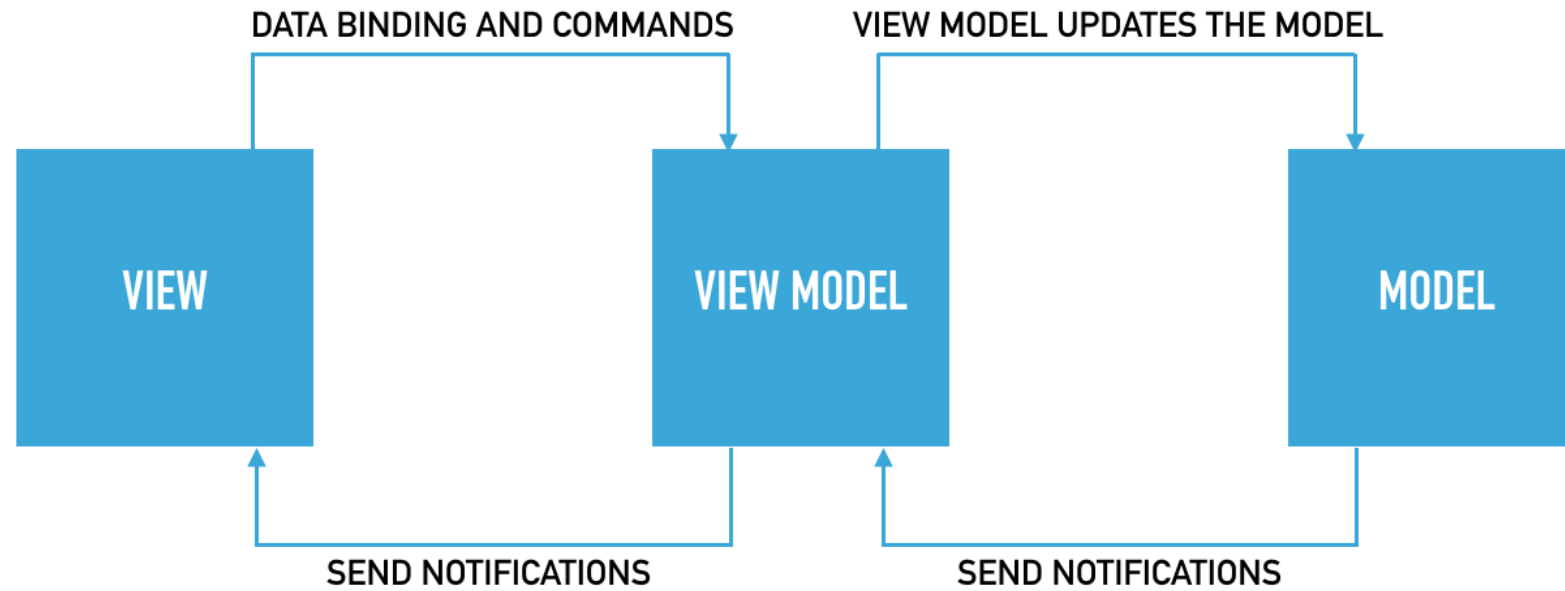
Google, Alibaba, Capital One, Tencent, eBay... are using Flutter



Flutter Architecture Overview



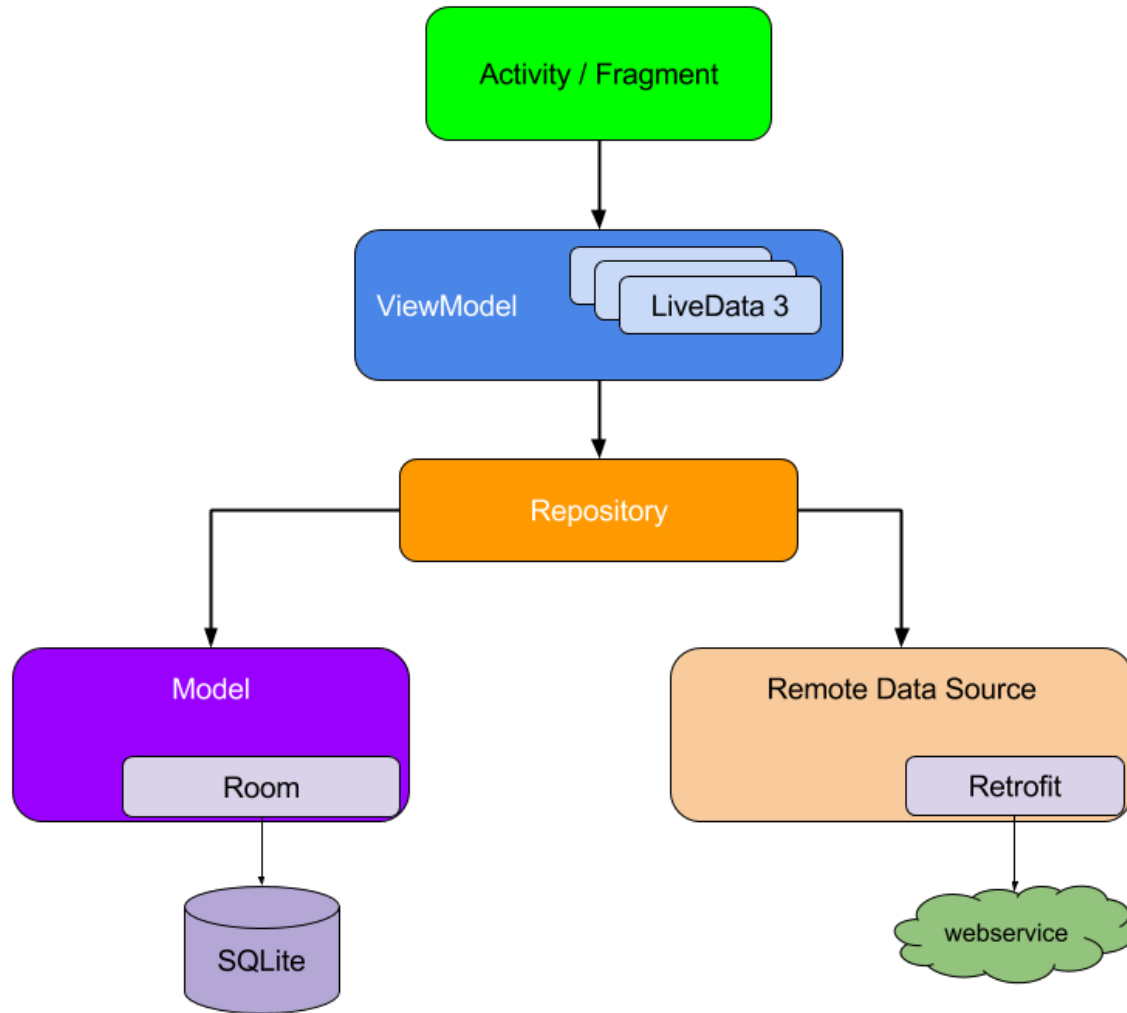
2. Flutter Architecture - MVVM



2. Flutter Application Architecture - Why MVVM?

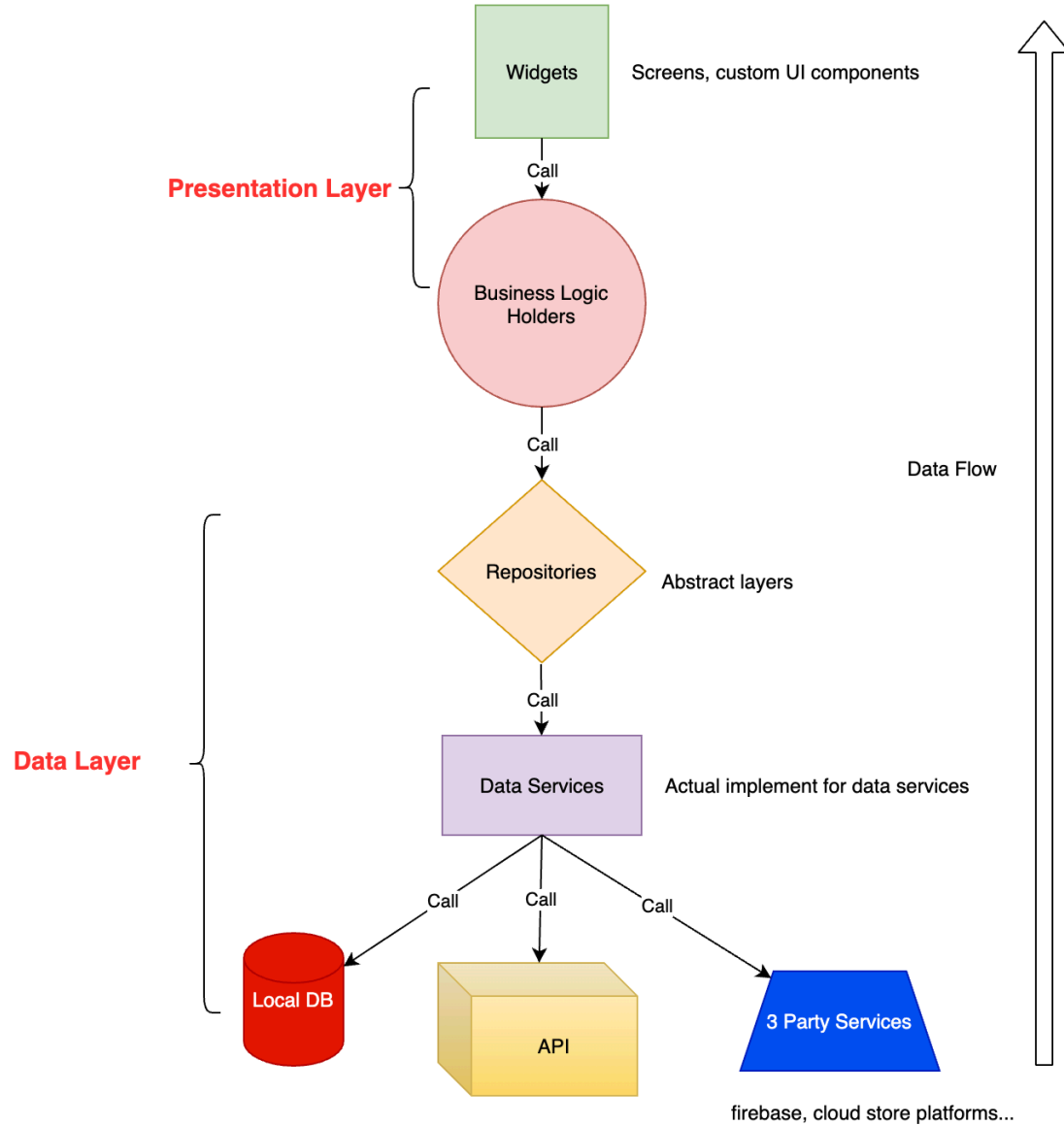
- ☑️ Avoid big controllers
- ☑️ Separation of concerns (UI and Business Logic)
- ☑️ Testability

2. Flutter Application Architecture - Repository Pattern

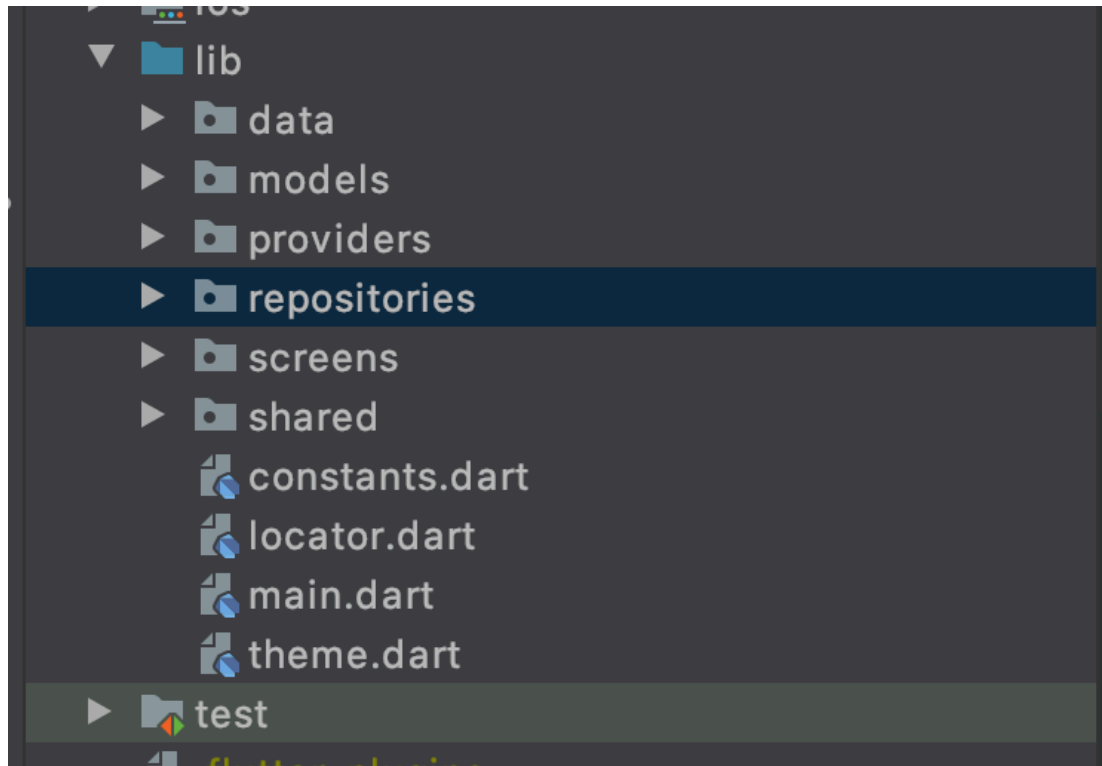


Decoupling the ways to access databases from the view model/ business logic.

2. Flutter Application Architecture



2. Flutter Application Architecture - Folder structure



2. Flutter Application Architecture

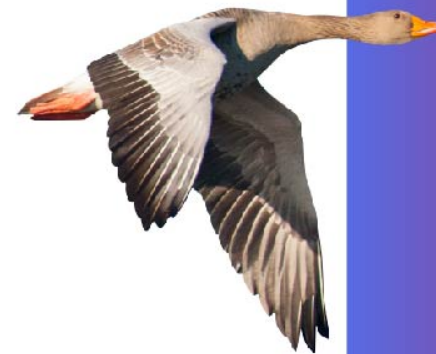
Business logic holder

- ☑ **Provider**
- ☑ riverPod
- ☑ **BLoC Pattern**
- ☑ ScopedModel
- ☑ MobX
- ☑ Redux



3. State management solutions

- ☑ Provider
- ☑ Redux
- ☑ BLoC
- ☑ setState
- ☑ InheritedWidget
- ☑ MobX



Why state management master?

- ✓ **State** is simply the data that represents the UI.
- ✓ **State management** is how we create, access, update and dispose this data.

Declarative UI

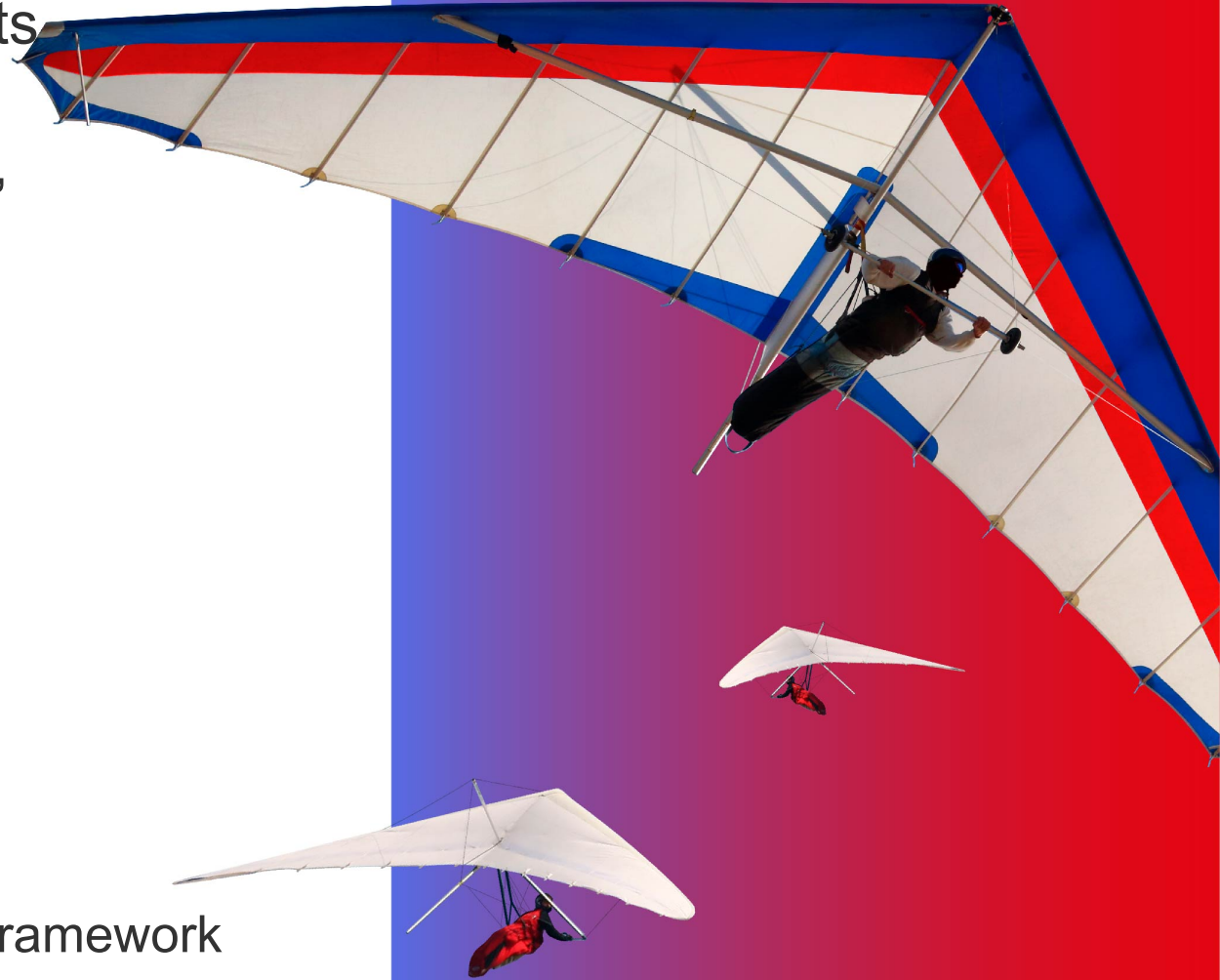
$$\text{UI} = f(\text{state})$$

The layout
on the screen

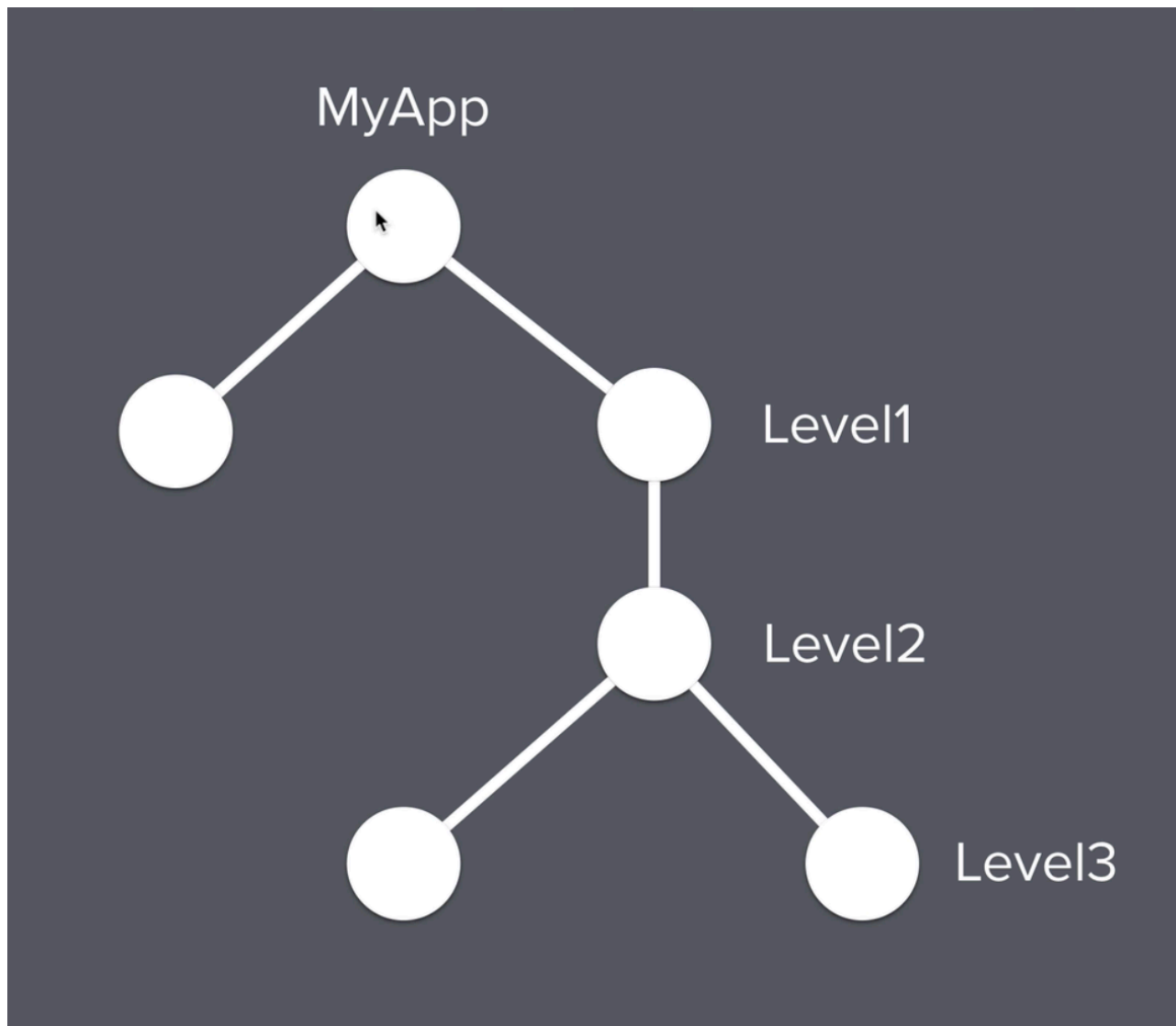
Your
build
methods

The application state

Flutter and React Native are declarative UI framework



Prop Drilling



Ephemeral state and app state

- ☑ Current page number of pageView
- ☑ Current selected item of dropdown list
- ☑ Current string value of Input text field.

...

- ☑ User preferences
- ☑ Login info
- ☑ Data which sharing multiple screens

...



Provider

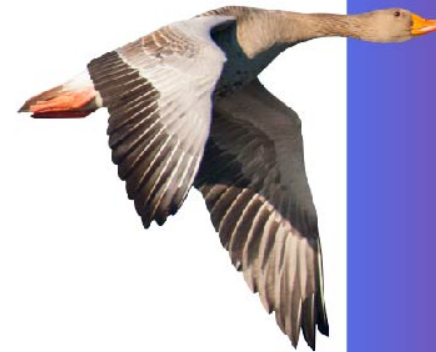
A wrapper around **InheritedWidget**

- ☑ Easy to use and reusable
- ☑ Reduce boilerplate
- ☑ Lazy-loading
- ☑ Common way to consume data(Provider.of, Consumer)
- ☑ Simplify allocation/disposal of resources



Provider and ScopedModel

- ☒ Wrapper around InheritedWidget
 - ☐ Provider has more options to implement
 - ☐ Use ChangeNotifier which is part of the Flutter framework
- Provider is ScopedModel 2.0



Provider - Common use classes

ChangeNotifier: model classes access to notifyListeners()

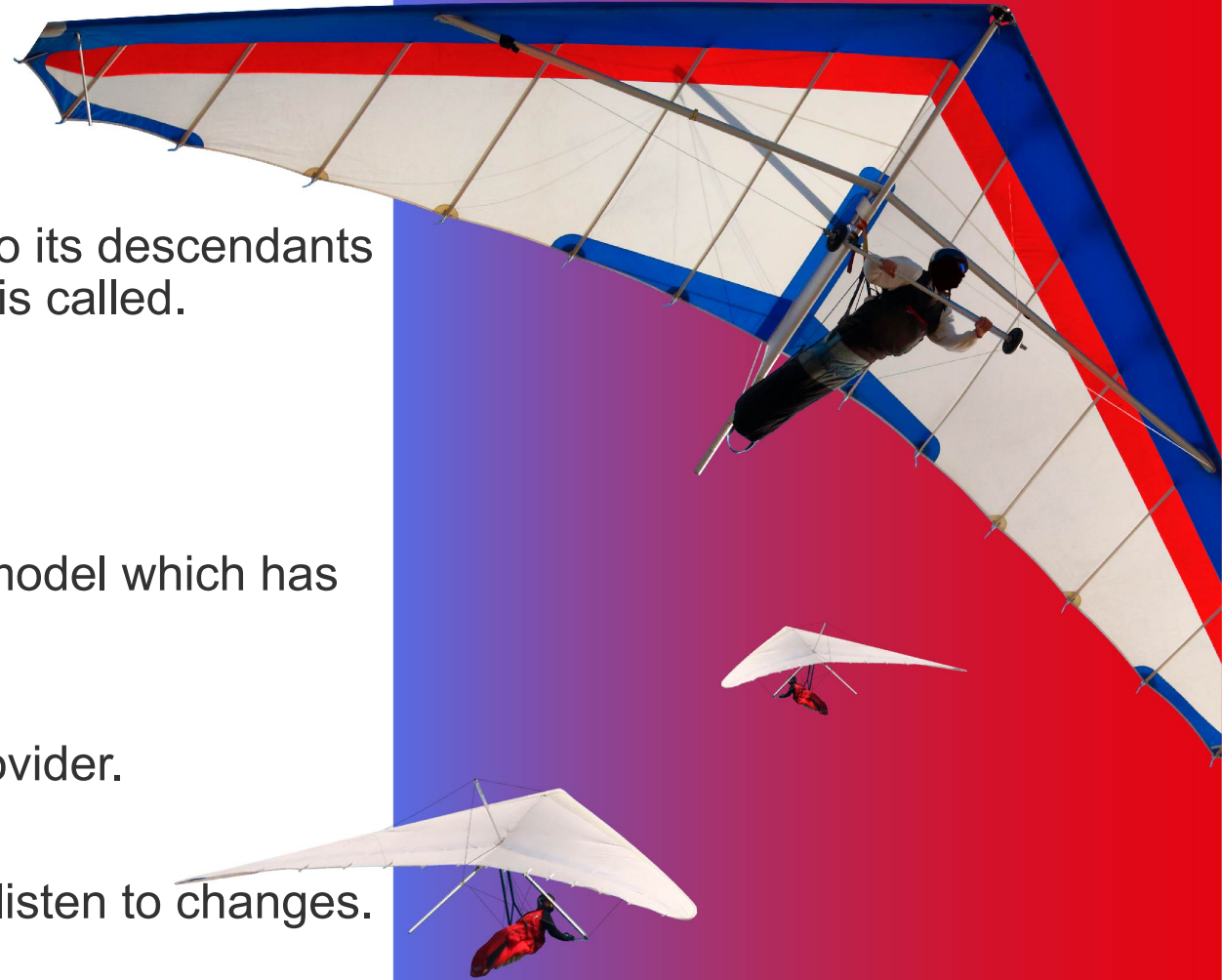
ChangeNotifierProvider: expose changeNotifier to its descendants and rebuild dependents when ever notifyListeners is called.

MultiProvider: provide multi models.

ProxyProvider: help inject one model to another model which has depends each other.

Consumer: consume data from ChangeNotifierProvider.

Provider.of: access model objects with or without listen to changes.



Provider - Important note

- ✓ The data inside the provider object should be private and can only modify inside. Otherwise you can't notify your listeners about the changes.
- ✓ Provider.Of method actually have the second parameter call "listen" to determine that you want to listen the changes or not. (for ex: you only want to call method and not interest to the data).



Provider

Demo

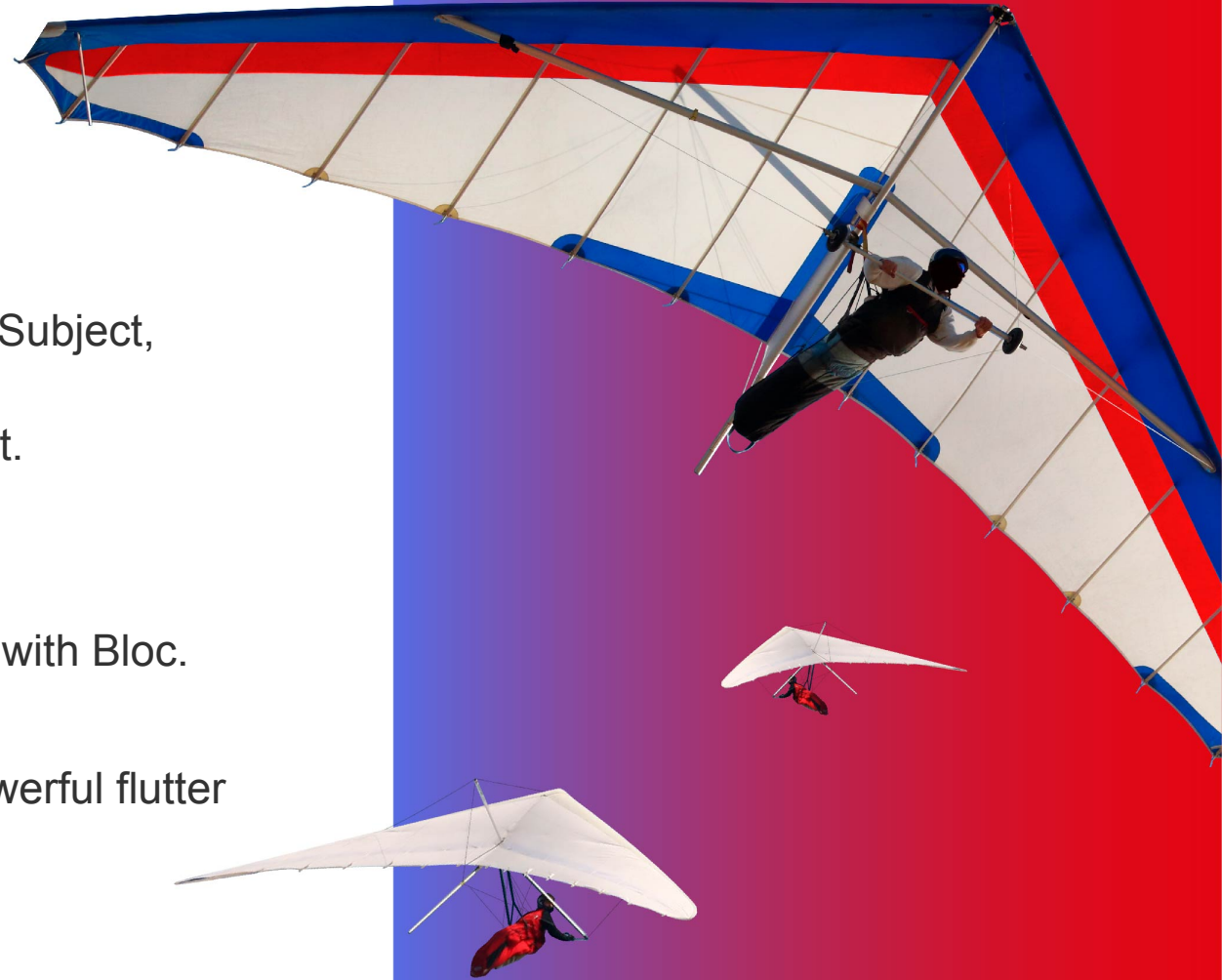
BLoC Pattern (Business Logic Components)

Old Implementation

- ✓ Created by Google and announced at Google IO'18.
- ✓ Use Reactive programming to handle flow of data.
- ✓ Streams, StreamController from dart:async, or PublishSubject, BehaviorSubject via the rxdart.
- ✓ No way to dispose stream controller in stateless widget.

Newest way to Implement BLoC (Preferred)

- ✓ Implement BLoC with Redux style (events, reducer...) with Bloc.
- ✓ Abstracts reactive aspects of the patterns using Cubit.
- ✓ Includes bloc package as core and flutter_bloc is a powerful flutter widgets to work with bloc.
- ✓ Use Provider internally to pass down bloc objects.



BLoC Pattern

- ✓ Stream based approach.
- ✓ Use Provider internally to pass down bloc objects.
- ✓ Business logic will be encapsulated to bloc objects.
- ✓ Testability.
- ✓ Separate of concerns.



Provider

Demo

BLoC and Provider, Which one to choose?

- ✓ Structure, clean and scaleable codes.
- ✓ Suitable for large project with big team.
- ✓ Can handle for complexed business logic.
- ✓ Testability.
- ✓ Nice documentation.
- ✓ Not suitable for small/simple app.
- ✓ More boilerplate codes.

- ✓ Easy, quick implementation.
- ✓ Less boilerplate codes.
- ✓ May suitable for small and medium projects.
- ✓ Testability.
- ✓ Provider is just a tool to manage states, access sharing data from widget tree, so we need to take extra care for the architecture.
- ✓ Calling Provider.of in a wrong way may cause bugs.

Provider itself is not a completed solution for state management, it likes a way to pass data down widgets tree by wrapper around InheritedWidget, make them easy to use and more useable. Or soft-of like Dependency Injection.

How Flutter community talk about this topic?

https://www.reddit.com/r/FlutterDev/comments/bmrvey/so_is_provider_recommended_over_bloc_just_watched/

One of the best explanation about Provider is here

So how do you manage the lifecycle of your BLoC object in a Flutter widget. You use a StatefulWidget. And what if you have to make your BloC object obtainable by multiple widgets. You lift state up:

1. Create a StatefulWidget at the top of the widget tree, so it's State can manage the lifecycle of your BloC object (in State.initState and State.dispose). Let's call the widget BlocProvider, it's only job is to manage the lifecycle of your object.
2. In the State.build method return an InheritedWidget that references your BloC object. Now all widgets in the subtree can obtain this widget (and thus the reference) using `context.inheritFromWidgetOfExactType` or `context.ancestorInheritedElementForWidgetOfExactType`.

This is what Provider does essentially. You can use it with BloC or without it, it's just a dependency injection library (that only works with Flutter widgets) that makes lifting state up in the widget tree more convenient.

More References

Declarative UI

<https://flutter.dev/docs/development/data-and-backend/state-mgmt/declarative>

<https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-other-frameworks-libraries-and-tools>

State management

<https://flutter.dev/docs/development/data-and-backend/state-mgmt>

<https://medium.com/flutter-community/making-sense-all-of-those-flutter-providers-e842e18f45dd>

<https://medium.com/flutter-nyc/a-closer-look-at-the-provider-package-993922d3a5a5> ([Provider](#))

<https://blog.codemagic.io/flutter-tutorial-provider/>

<https://bloclibrary.dev/#/gettingstarted> ([BLoC new](#))

<https://www.freecodecamp.org/news/how-to-handle-state-in-flutter-using-the-bloc-pattern-8ed2f1e49a13/> ([BLoC old](#))

<https://blog.codemagic.io/flutter-tutorial-pros-and-cons-of-state-management-approaches/>

Architecture

<https://www.filledstacks.com/post/flutter-provider-architecture-sharing-data-across-your-models/>

<https://resocoder.com/flutter-clean-architecture-tdd/>

Packages

Provider: <https://pub.dev/packages/provider>

BLoC:

<https://pub.dev/packages/bloc>

https://pub.dev/packages/flutter_bloc

DI: https://pub.dev/packages/get_it



Q&A



Thank you!