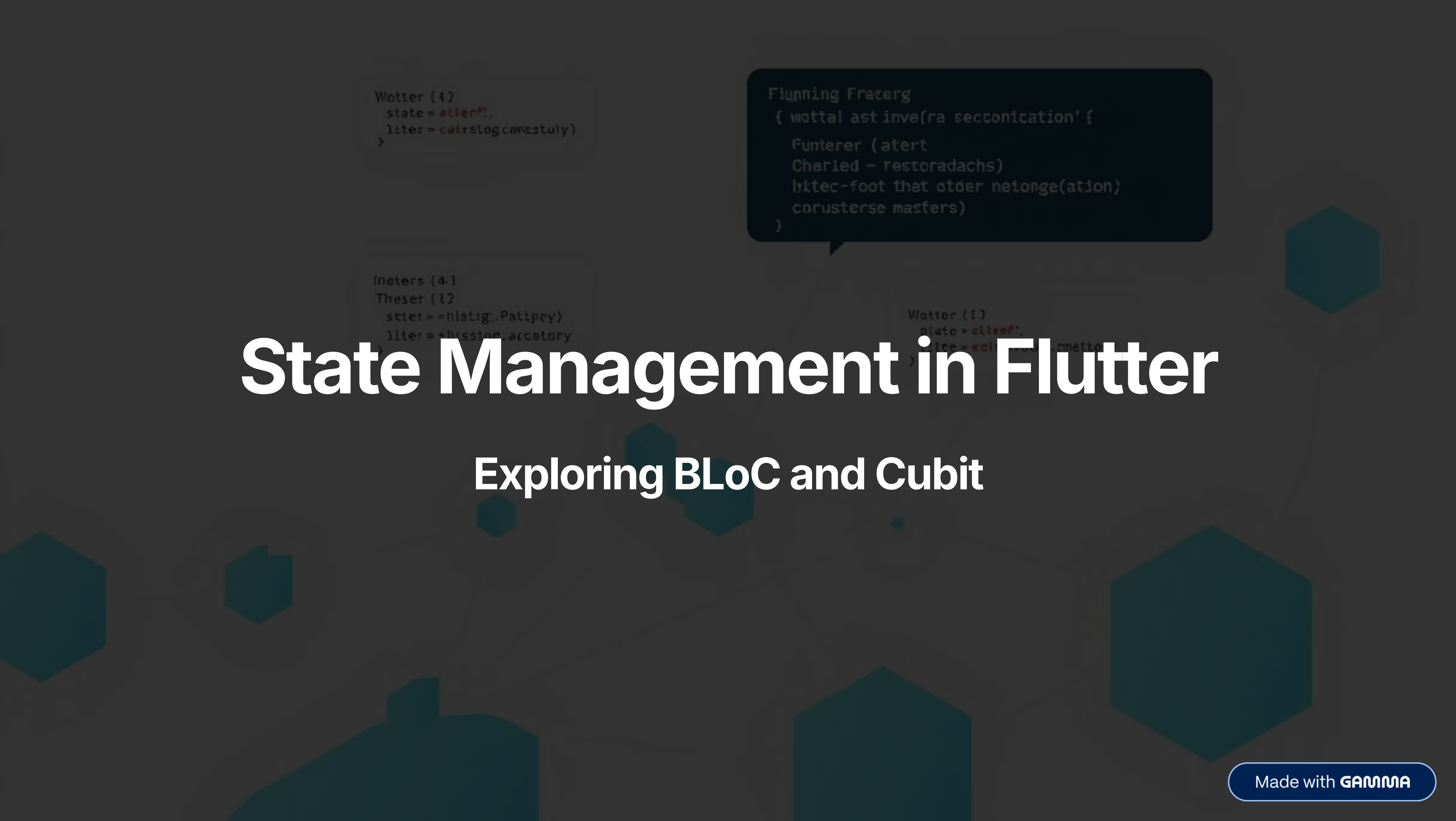


State Management in Flutter

Exploring BLoC and Cubit

Understanding State Management in Flutter

What is State Management?

process of **handling the data (state) of your application.**

Why Not setState Management?

`setState()` rebuilds the **entire widget**, even if only one part of the UI changed.

This can lead to **performance issues**, especially in complex UIs.

Why BLoC and Cubit?

BLoC (Business Logic Component)

BLoC helps separate **business logic from UI** using events and states. It provides a robust, predictable, and testable way to manage application state.

Cubit

Cubit is a simpler, lighter version of BLoC. It uses functions instead of events to emit new states, making it ideal for **less complex state management scenarios**.

Understanding The BLoC Package

Events

Events are the inputs to a BLoC. They represent user interactions or other triggers that signal a change is needed.

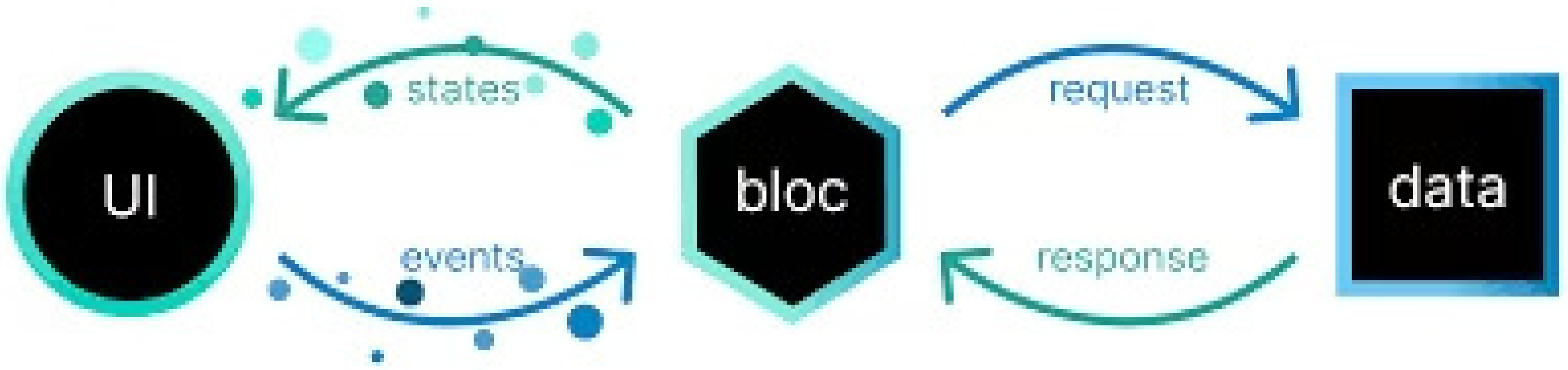
States

States are the immutable outputs of a BLoC.

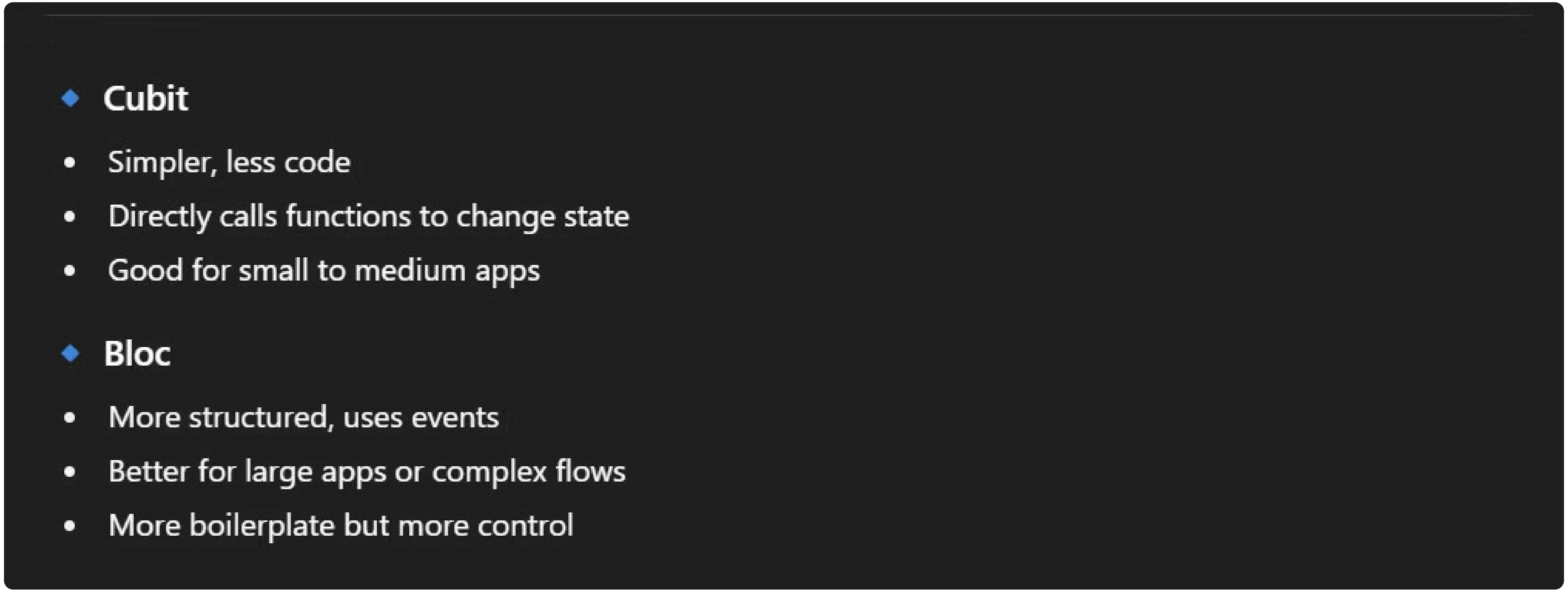
BLoC

The BLoC (Business Logic Component) processes incoming events and transforms them into new states, separating UI from logic.

The BLoC package provides a robust framework for managing application state in Flutter, emphasizing separation of concerns and testability by using distinct Events, BLoCs, and States. This architecture ensures predictable state changes and makes debugging straightforward.


states
UI
events
bloc
request
response
data

bloc vs cubit



◆ **Cubit**

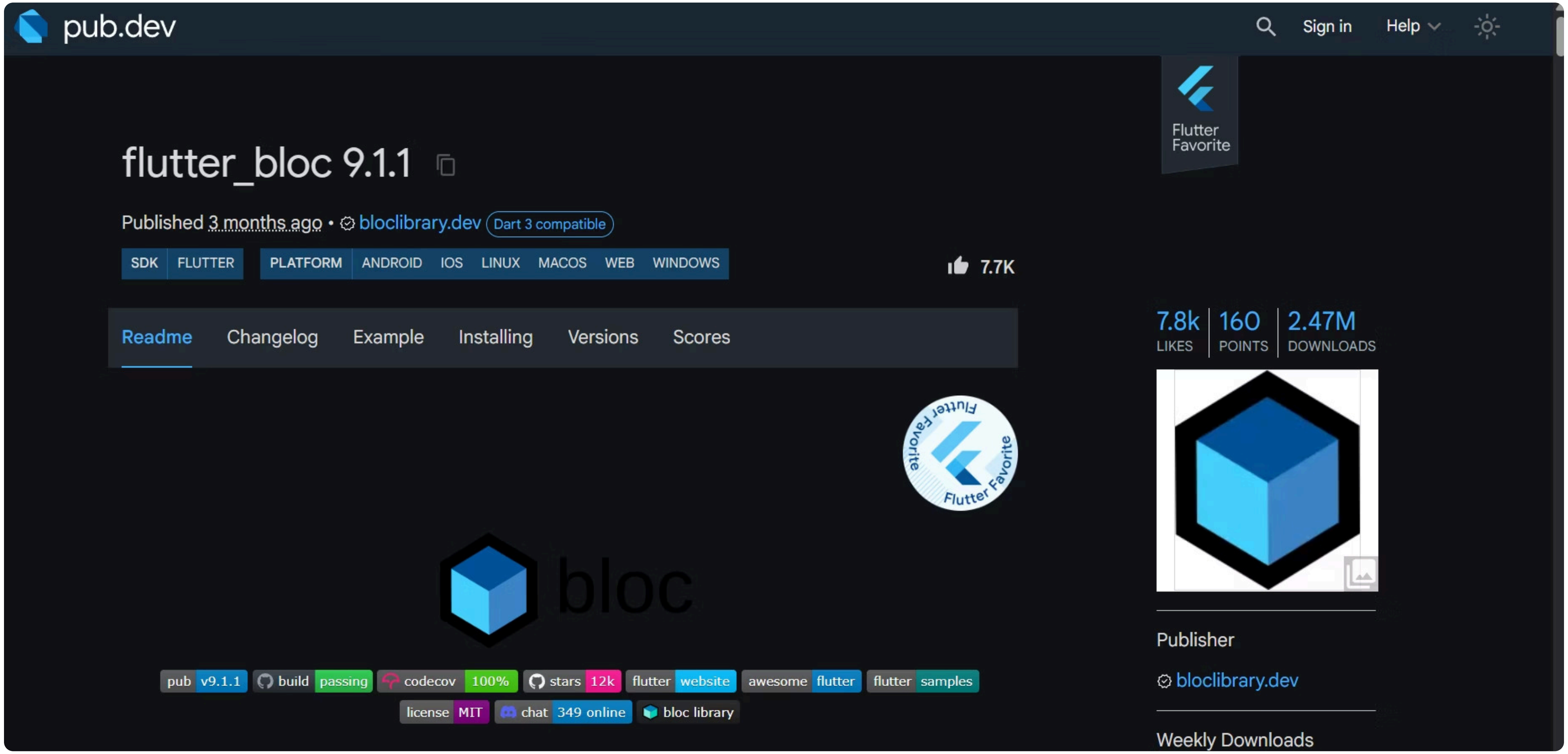
- Simpler, less code
- Directly calls functions to change state
- Good for small to medium apps

◆ **Bloc**

- More structured, uses events
- Better for large apps or complex flows
- More boilerplate but more control

counter app using SetState

flutter_bloc package



pub.dev

Sign in Help

Flutter Favorite

flutter_bloc 9.1.1

Published 3 months ago • bloclibrary.dev Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

7.7K

Readme Changelog Example Installing Versions Scores

7.8k	160	2.47M
LIKES	POINTS	DOWNLOADS

bloc

pub v9.1.1 build passing codecov 100% stars 12k flutter website awesome flutter flutter samples
license MIT chat 349 online bloc library

Publisher

bloclibrary.dev

Weekly Downloads

Made with GAMMA

How to Use Cubit: A Step-by-Step Flow

1. Install `flutter_bloc`

Add the `flutter_bloc` package to your `pubspec.yaml` dependencies to enable Cubit functionality.

2. Define Cubit Files

Create a dedicated folder containing two files: one for the Cubit's state (e.g., `counter_state.dart`) and one for the Cubit logic itself (e.g., `counter_cubit.dart`).

3. Design Your Cubit State

Define an abstract base class for your Cubit's states, and then create immutable concrete state classes that extend it.

4. Implement Cubit Logic

Create your Cubit by extending `Cubit`. Define methods that use `emit()` to broadcast new states in response to actions.

5. Integrate with UI

Provide your Cubit to the widget tree using `BlocProvider`. Use `BlocBuilder` or `BlocListener` in your UI to consume and react to state changes.

how to connect between ui and cubit ??

Steps:

1. Provide the Cubit:

```
return BlocProvider(
  create: (_) => CounterCubit(),
  child: CounterScreen(), // Your UI screen
);
```

2. Build UI with Cubit State:

```


BlocBuilder<CounterCubit, CounterState>(


builder: (context, state)
{


    return Column([]);
},


)
```

Task:

Build a simple Flutter app using **Cubit** that allows the user to type their name in a **TextField**, and displays the typed name live on the screen below the input.

Hint=> use onChanged function inside TextField.

task

Favorite Items App –

1. Show a list of items in a `ListView`.
2. Each item has a star icon (favorite toggle).
3. On tap → toggle favorite state via **Cubit**.
4. UI updates automatically when state changes.