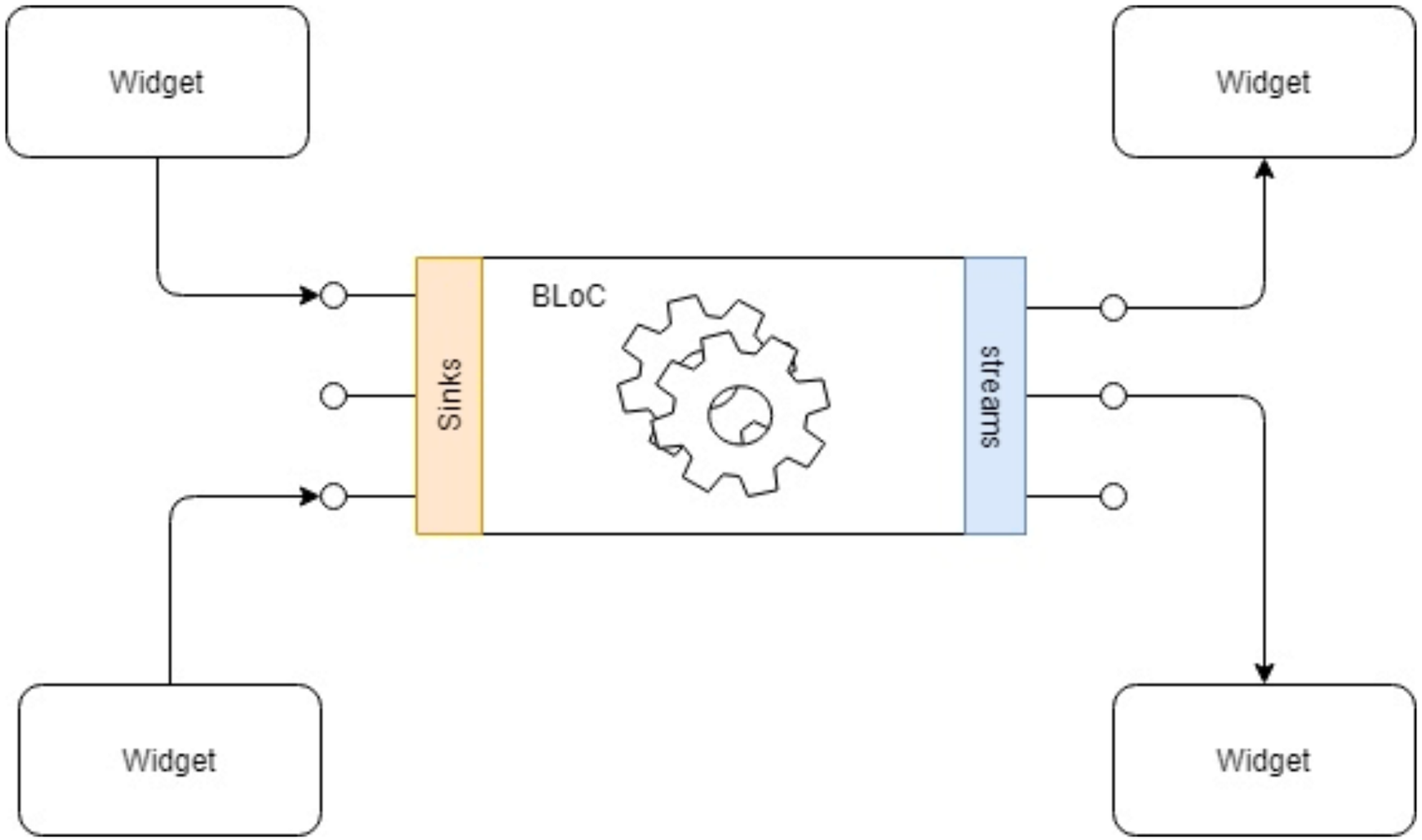
Bloc Library: Basics & Beyond 🚀

Felix Angelov @ Very Good Ventures

Very Good Ventures, Chicago 🕶🦄

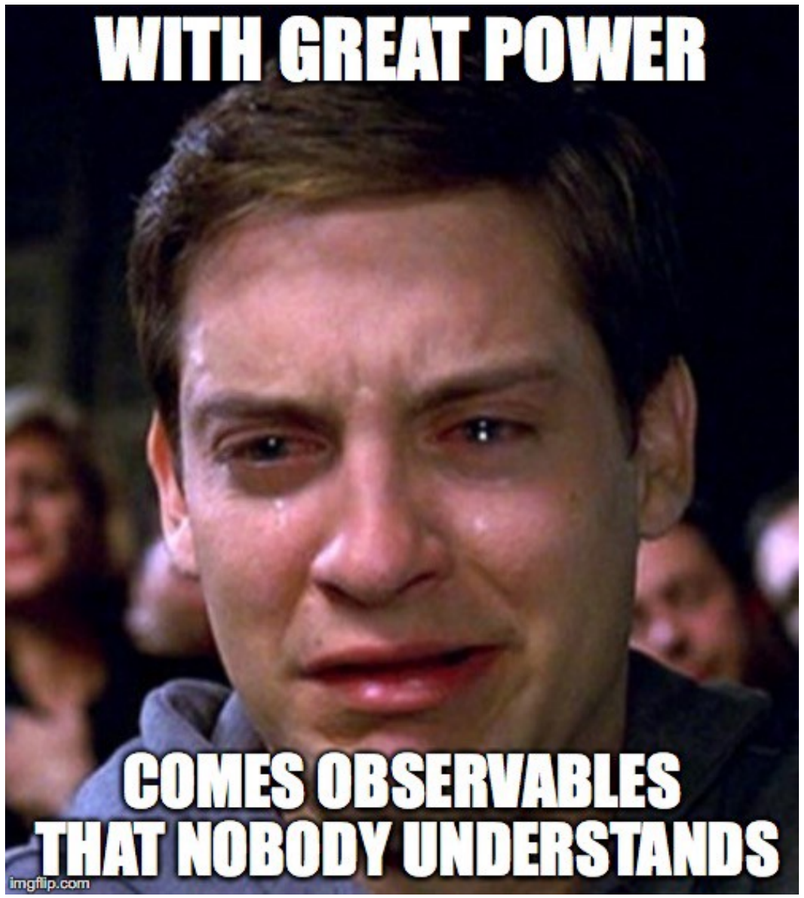
about me

In the beginning...



https://www.didierboelens.com/2018/08/reactive-programming---streams---bloc/

After a while...



https://medium.com/@drm/unsubscribe-from-angular-observables-2102dad5b732

Goals

- ☐ decouple UI & business logic
- ☐ easy to test
- ☐ predictable/maintainable

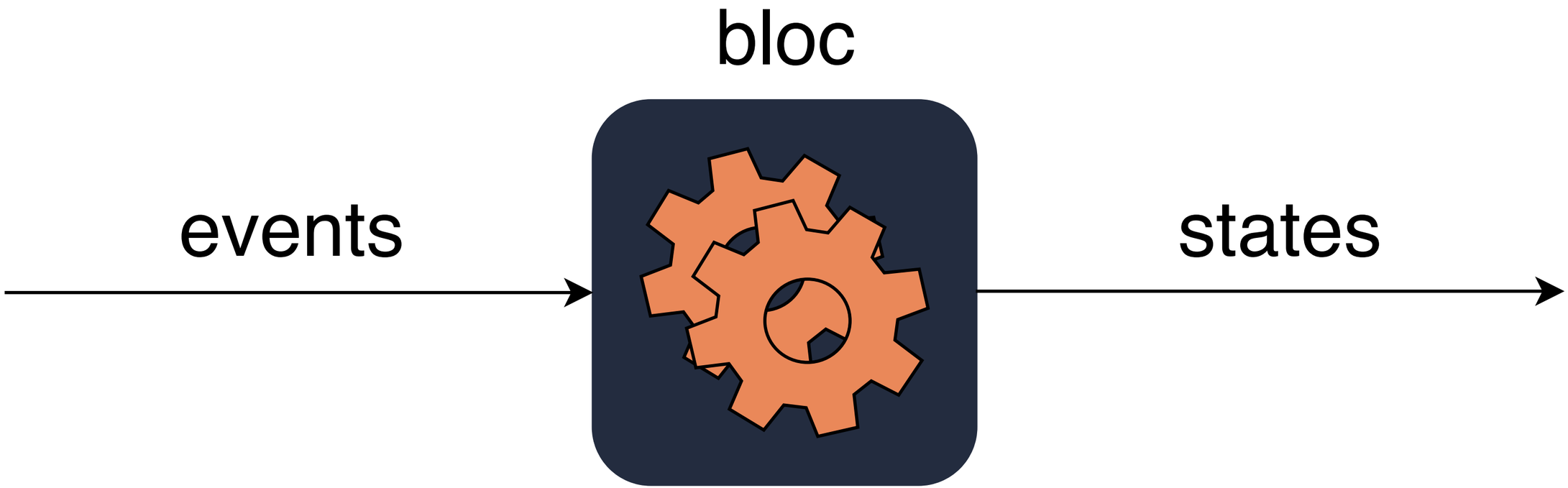
bloc

A predictable state management library that helps implement the BLoC (Business Logic Component) design pattern.

v 3.0.0 • updated: Dec 25, 2019 • bloclibrary.dev DART FLUTTER

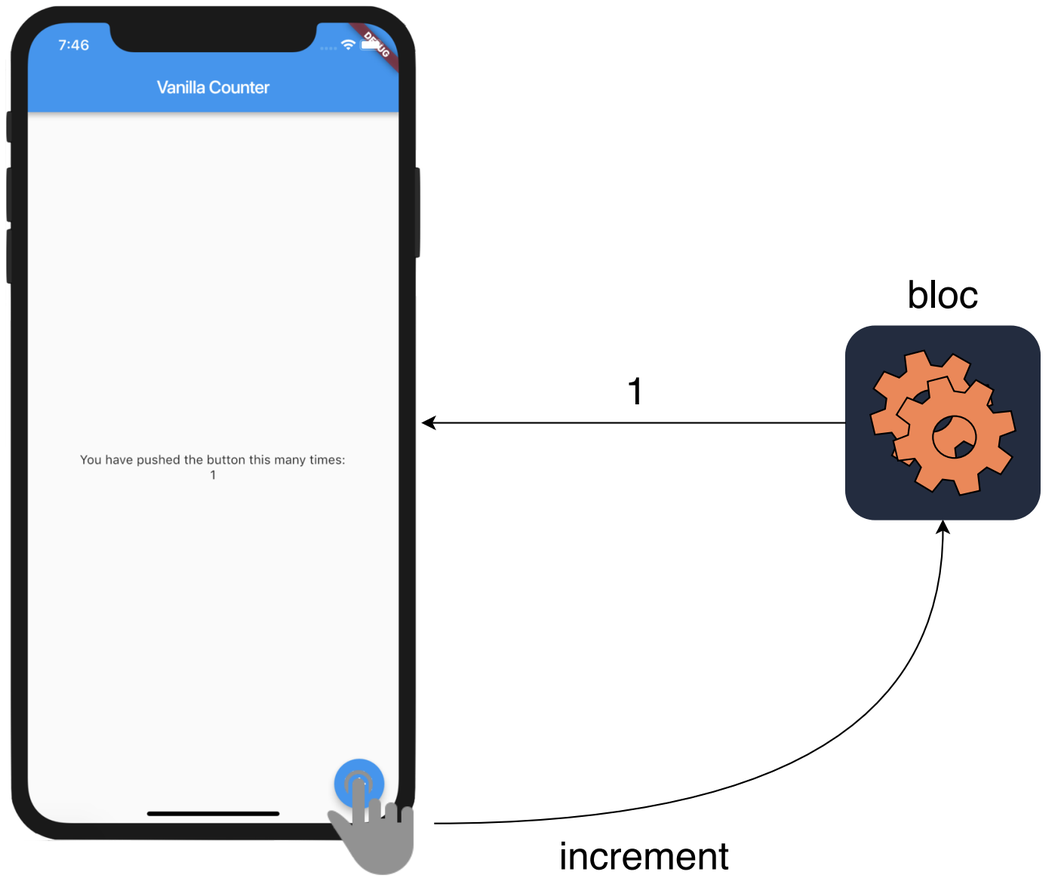
100

Overview



A bloc converts incoming events into outgoing states.

Counter Overview



package:bloc

```
name: counter_bloc
description: A counter bloc example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  bloc: ^3.0.0
```

package:bloc

```
name: counter_bloc
description: A counter bloc example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  bloc: ^3.0.0
```

Bloc Anatomy

```
import 'dart:async';
import 'package:bloc/bloc.dart';
```

Bloc Anatomy

```dart
import 'dart:async';
import 'package:bloc/bloc.dart';

enum CounterEvent { increment, decrement }
```

Bloc Anatomy

```dart
import 'dart:async';
import 'package:bloc/bloc.dart';

enum CounterEvent { increment, decrement }

class CounterBloc extends Bloc<CounterEvent, int> {}
```

Bloc Anatomy

```dart
import 'dart:async';
import 'package:bloc/bloc.dart';

enum CounterEvent { increment, decrement }

class CounterBloc extends Bloc<CounterEvent, int> {
    @override
    int get initialState => 0;
}
```

Bloc Anatomy

```
...

class CounterBloc extends Bloc<CounterEvent, int> {
    ...

    @override
    Stream<int> mapEventToState(CounterEvent event) async* {

    }
}
```

Bloc Anatomy

```
...

class CounterBloc extends Bloc<CounterEvent, int> {
    ...

    @override
    Stream<int> mapEventToState(CounterEvent event) async* {
        switch (event) {

        }
    }
}
```

Bloc Anatomy

```
...

class CounterBloc extends Bloc<CounterEvent, int> {
    ...

    @override
    Stream<int> mapEventToState(CounterEvent event) async* {
        switch (event) {
            case CounterEvent.increment:
                yield state + 1;
                break;
        }
    }
}
```

Bloc Anatomy

```
...

class CounterBloc extends Bloc<CounterEvent, int> {
    ...

    @override
    Stream<int> mapEventToState(CounterEvent event) async* {
        switch (event) {
            case CounterEvent.increment:
                yield state + 1;
                break;
            case CounterEvent.decrement:
                yield state - 1;
                break;
        }
    }
}
```

Bloc Anatomy

```dart
import 'dart:async';
import 'package:bloc/bloc.dart';

enum CounterEvent { increment, decrement }

class CounterBloc extends Bloc<CounterEvent, int> {
    @override
    int get initialState => 0;

    @override
    Stream<int> mapEventToState(CounterEvent event) async* {
        switch (event) {
            case CounterEvent.increment:
                yield state + 1;
                break;
            case CounterEvent.decrement:
                yield state - 1;
                break;
        }
    }
}
```

Bloc in Action

```dart
import 'package:counter_bloc/counter_bloc.dart';

void main() {
    final counterBloc = CounterBloc();

    counterBloc.listen(print);

    counterBloc.add(CounterEvent.increment);
    counterBloc.add(CounterEvent.decrement);
}
```

Bloc in Action

```
import 'package:counter_bloc/counter_bloc.dart';

void main() {
    final counterBloc = CounterBloc();

    counterBloc.listen(print);

    counterBloc.add(CounterEvent.increment);
    counterBloc.add(CounterEvent.decrement);
}
```

Bloc in Action

```dart
import 'package:counter_bloc/counter_bloc.dart';

void main() {
    final counterBloc = CounterBloc();

    counterBloc.listen(print);

    counterBloc.add(CounterEvent.increment);
    counterBloc.add(CounterEvent.decrement);
}
```

Bloc in Action

```
import 'package:counter_bloc/counter_bloc.dart';

void main() {
    final counterBloc = CounterBloc();

    counterBloc.listen(print);

    counterBloc.add(CounterEvent.increment);
    counterBloc.add(CounterEvent.decrement);
}
```

Bloc in Action

```dart
import 'package:counter_bloc/counter_bloc.dart';

void main() {
    final counterBloc = CounterBloc();

    counterBloc.listen(print);

    counterBloc.add(CounterEvent.increment);
    counterBloc.add(CounterEvent.decrement);
}
```

Bloc in Action

```
$ dart example/main.dart
0
1
0
```

Bloc in Action

```
$ dart example/main.dart
0 // initialState
1
0
```

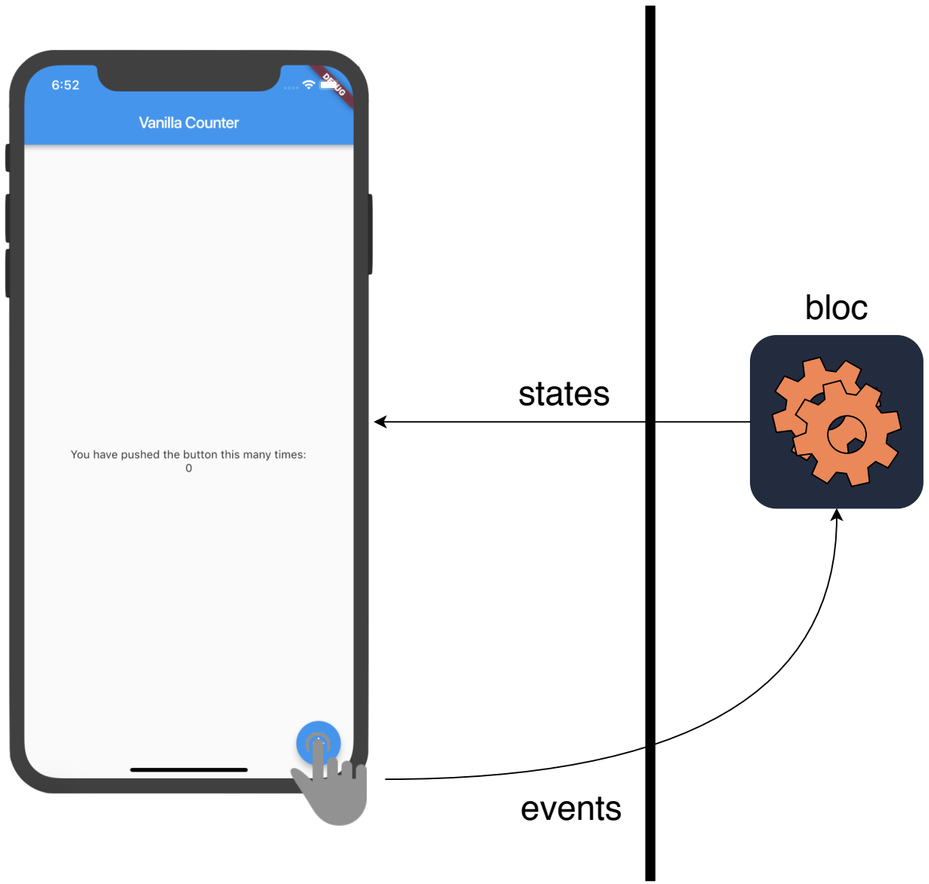
Bloc in Action

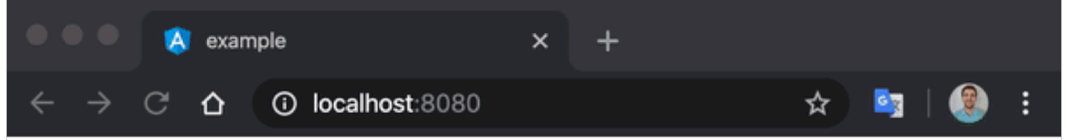
```
$ dart example/main.dart
0
1 // CounterEvent.increment
0
```

Bloc in Action

```
$ dart example/main.dart
0
1
0 // CounterEvent.decrement
```

Separation of Concerns

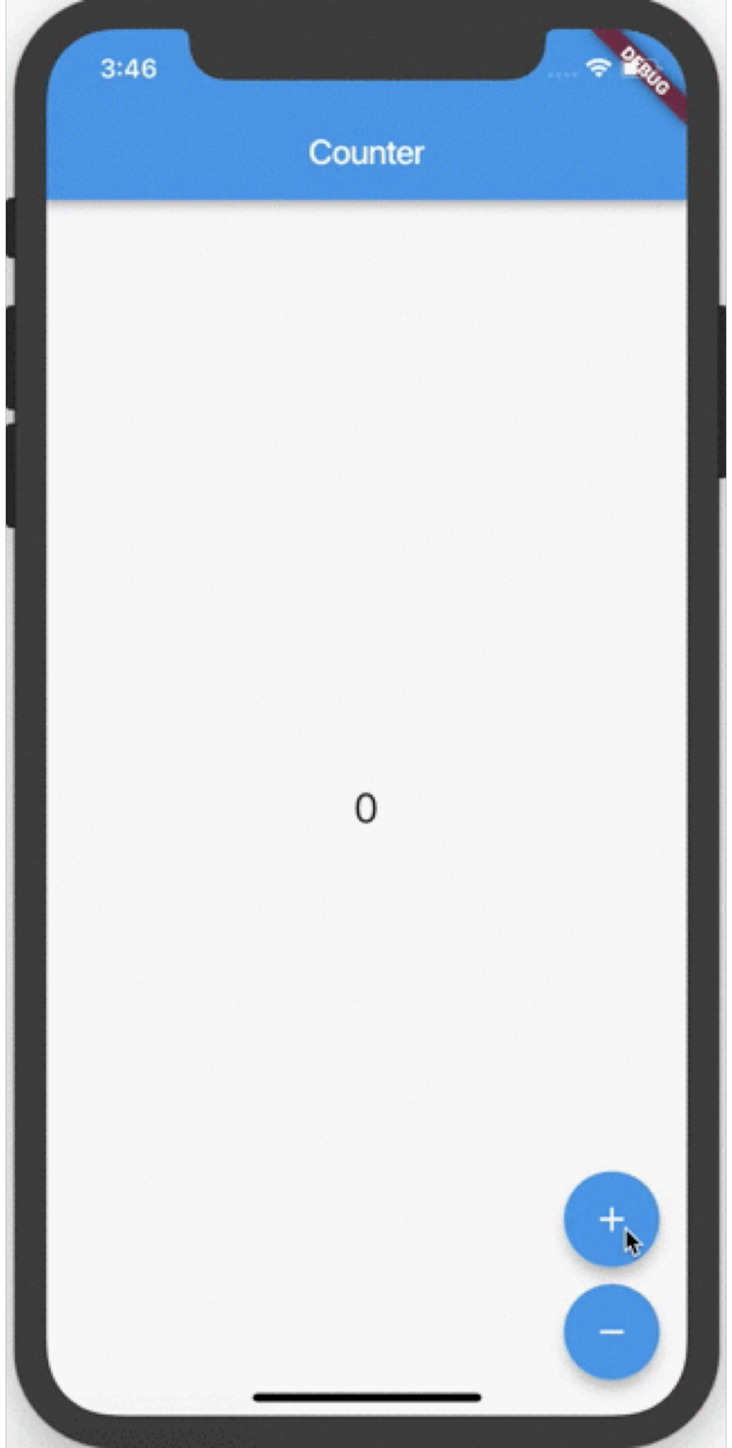




Counter App

Current Count: 0

+ -



Async?

Async Bloc

```
abstract class CounterState {}

class CounterLoadInProgress extends CounterState {}

class CounterLoadSuccess extends CounterState {
  CounterLoadSuccess(this.count);

  final int count;
}

class CounterLoadFailure extends CounterState {}
```

Async Bloc

```
abstract class CounterState {}

class CounterLoadInProgress extends CounterState {}

class CounterLoadSuccess extends CounterState {
  CounterLoadSuccess(this.count);

  final int count;
}

class CounterLoadFailure extends CounterState {}
```

Async Bloc

```
abstract class CounterState {}

class CounterLoadInProgress extends CounterState {}

class CounterLoadSuccess extends CounterState {
  CounterLoadSuccess(this.count);

  final int count;
}

class CounterLoadFailure extends CounterState {}
```

Async Bloc

```
abstract class CounterState {}

class CounterLoadInProgress extends CounterState {}

class CounterLoadSuccess extends CounterState {
  CounterLoadSuccess(this.count);

  final int count;
}

class CounterLoadFailure extends CounterState {}
```

Async Bloc

```
...

class CounterBloc extends Bloc<CounterEvent, CounterState> {...}
```

Async Bloc

```
...

class CounterBloc extends Bloc<CounterEvent, CounterState> {
    CounterBloc(this._counterService);

    final CounterService _counterService;

    ...
}
```

Async Bloc

```
...

@override
int get initialState => CounterLoadInProgress();

@override
Stream<int> mapEventToState(CounterEvent event) async* {
    switch (event) {
        case CounterEvent.increment:
            yield* _mapIncrementToState();
            break;
        case CounterEvent.decrement:
            yield* _mapDecrementToState();
            break;
    }
}
```

Async Bloc

```
...

@override
int get initialState => CounterLoadInProgress();

@override
Stream<int> mapEventToState(CounterEvent event) async* {
    switch (event) {
        case CounterEvent.increment:
            yield* _mapIncrementToState();
            break;
        case CounterEvent.decrement:
            yield* _mapDecrementToState();
            break;
    }
}
```

Async Bloc

```
...

Stream<CounterState> _mapIncrementToState() async* {
  yield CounterLoadInProgress();
  try {
    final asyncCount = await _counterService.increment();
    yield CounterLoadSuccess(asyncCount);
  } catch (_) {
    yield CounterLoadFailure();
  }
}
```

Async Bloc

```
...

Stream<CounterState> _mapIncrementToState() async* {
  yield CounterLoadInProgress();
  try {
    final asyncCount = await _counterService.increment();
    yield CounterLoadSuccess(asyncCount);
  } catch (_) {
    yield CounterLoadFailure();
  }
}
```

Async Bloc

```
...

Stream<CounterState> _mapIncrementToState() async* {
  yield CounterLoadInProgress();
  try {
    final asyncCount = await _counterService.increment();
    yield CounterLoadSuccess(asyncCount);
  } catch (_) {
    yield CounterLoadFailure();
  }
}
```

Async Bloc

```
...

Stream<CounterState> _mapIncrementToState() async* {
  yield CounterLoadInProgress();
  try {
    final asyncCount = await _counterService.increment();
    yield CounterLoadSuccess(asyncCount);
  } catch (_) {
    yield CounterLoadFailure();
  }
}
```

Async Bloc

```
...

Stream<CounterState> _mapDecrementToState() async* {
  yield CounterLoadInProgress();
  try {
    final asyncCount = await _counterService.decrement();
    yield CounterLoadSuccess(asyncCount);
  } catch (_) {
    yield CounterLoadFailure();
  }
}
```

Async Bloc

```
...

Stream<CounterState> _mapDecrementToState() async* {
  yield CounterLoadInProgress();
  try {
    final asyncCount = await _counterService.decrement();
    yield CounterLoadSuccess(asyncCount);
  } catch (_) {
    yield CounterLoadFailure();
  }
}
```

Goals

- ☑ **decouple UI & business logic**
- ☐ easy to test
- ☐ predictable/maintainable

Goals

- [x] decouple UI & business logic
- [ ] **easy to test**
- [ ] predictable/maintainable

bloc_test

A testing library which makes it easy to test blocs. Built to be used with the bloc state management package.

v 3.0.1 • updated: Jan 3, 2020 • bloclibrary.dev DART FLUTTER

95

package:bloc_test

```
name: counter_bloc
description: A counter bloc example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  bloc: ^3.0.0

dev_dependencies:
  test: ^1.11.1
  bloc_test: ^3.0.0
```

package:bloc_test

```
name: counter_bloc
description: A counter bloc example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  bloc: ^3.0.0

dev_dependencies:
  test: ^1.11.1
  bloc_test: ^3.0.0
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test Anatomy

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    blocTest(
      'emits [0] when no events are added',
      build: () => CounterBloc(),
      expect: [0],
    );
  });
}
```

bloc_test in Action

```
$ pub run test
✓ CounterBloc emits [0] when no events are added
00:01 +1: All tests passed!
```

bloc_test in Action

```
$ pub run test
✓ CounterBloc emits [0] when no events are added
00:01 +1: All tests passed!
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, 1] when CounterEvent.increment is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.increment),
      expect: [0, 1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, 1] when CounterEvent.increment is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.increment),
      expect: [0, 1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, 1] when CounterEvent.increment is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.increment),
      expect: [0, 1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, 1] when CounterEvent.increment is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.increment),
      expect: [0, 1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, 1] when CounterEvent.increment is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.increment),
      expect: [0, 1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, -1] when CounterEvent.decrement is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.decrement),
      expect: [0, -1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, -1] when CounterEvent.decrement is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.decrement),
      expect: [0, -1],
    );
  });
}
```

One More Time

```dart
import 'package:test/test.dart';
import 'package:bloc_test/bloc_test.dart';

import '../counter_bloc.dart';

void main() {
  group('CounterBloc', () {
    ...

    blocTest(
      'emits [0, -1] when CounterEvent.decrement is added',
      build: () => CounterBloc(),
      act: (counterBloc) => counterBloc.add(CounterEvent.decrement),
      expect: [0, -1],
    );
  });
}
```

All Tests in Action

```
$ pub run test
✓ CounterBloc emits [0] when no events are added
✓ CounterBloc emits [0, 1] when CounterEvent.increment is added
✓ CounterBloc emits [0, -1] when CounterEvent.decrement is added
00:01 +3: All tests passed!
```

All Tests in Action

```
$ pub run test
✓ CounterBloc emits [0] when no events are added
✓ CounterBloc emits [0, 1] when CounterEvent.increment is added
✓ CounterBloc emits [0, -1] when CounterEvent.decrement is added
00:01 +3: All tests passed!
```

Async Tests?

package:bloc_test

```
name: counter_bloc
description: A counter bloc example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  bloc: ^3.0.0

dev_dependencies:
  test: ^1.11.1
  bloc_test: ^3.0.0
  mockito: ^4.0.0
```

Async Bloc Tests

```
import 'package:mockito/mockito.dart';

class MockCounterService extends Mock implements CounterService {}

...
```

Async Bloc Tests

```
...

blocTest(
  'emits [CounterLoadInProgress(), CounterLoadSuccess(1)] when CounterEvent.increment is added',
  build: () {
    final counterService = MockCounterService();
    when(counterService.increment).thenAnswer((_) => Future.value(1));
    return CounterBloc(counterService);
  },
  act: (counterBloc) => counterBloc.add(CounterEvent.increment),
  expect: [CounterLoadInProgress(), CounterLoadSuccess(1)],
);
```

Async Bloc Tests

```
...

blocTest(
  'emits [CounterLoadInProgress(), CounterLoadSuccess(1)] when CounterEvent.increment is added',
  build: () {
    final counterService = MockCounterService();
    when(counterService.increment).thenAnswer((_) => Future.value(1));
    return CounterBloc(counterService);
  },
  act: (counterBloc) => counterBloc.add(CounterEvent.increment),
  expect: [CounterLoadInProgress(), CounterLoadSuccess(1)],
);
```

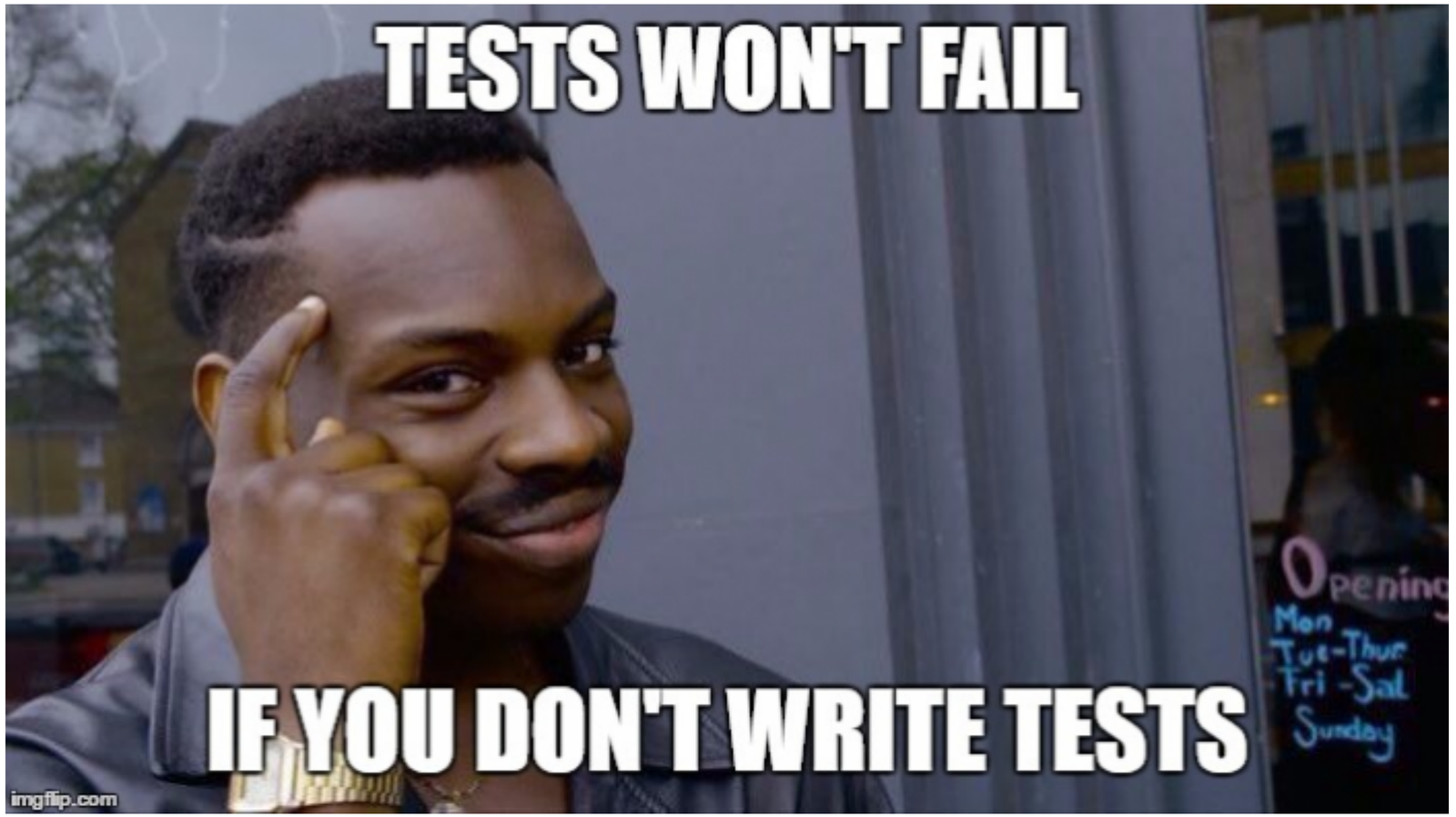
Async Bloc Tests

```
...

blocTest(
  'emits [CounterLoadInProgress(), CounterLoadSuccess(1)] when CounterEvent.increment is added',
  build: () {
    final counterService = MockCounterService();
    when(counterService.increment).thenAnswer((_) => Future.value(1));
    return CounterBloc(counterService);
  },
  act: (counterBloc) => counterBloc.add(CounterEvent.increment),
  expect: [CounterLoadInProgress(), CounterLoadSuccess(1)],
);
```

Async Bloc Tests

```
...

blocTest(
  'emits [CounterLoadInProgress(), CounterLoadSuccess(1)] when CounterEvent.increment is added',
  build: () {
    final counterService = MockCounterService();
    when(counterService.increment).thenAnswer((_) => Future.value(1));
    return CounterBloc(counterService);
  },
  act: (counterBloc) => counterBloc.add(CounterEvent.increment),
  expect: [CounterLoadInProgress(), CounterLoadSuccess(1)],
);
```

Not You


TESTS WON'T FAIL
IF YOU DON'T WRITE TESTS
imgflip.com
Opening
Mon
Tue-Thur
Fri -Sat
Sunday

Goals

- ☑ decouple UI & business logic
- ☑ **easy to test**
- ☐ predictable/maintainable

Goals

- ☑ decouple **UI** & business logic
- ☑ easy to test
- ☐ predictable/maintainable

flutter_bloc

Flutter Widgets that make it easy to implement the BLoC (Business Logic Component) design pattern. Built to be used with the bloc state management package.

100

v 3.1.0 • updated: Jan 1, 2020 • bloclibrary.dev FLUTTER

package:flutter_bloc

```
name: counter_app
description: A counter app example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_bloc: ^3.0.0
  counter_bloc:
    path: ../counter_bloc

flutter:
  uses-material-design: true
```

package:flutter_bloc

```
name: counter_app
description: A counter app example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_bloc: ^3.0.0
  counter_bloc:
    path: ../counter_bloc

flutter:
  uses-material-design: true
```

package:flutter_bloc

```
name: counter_app
description: A counter app example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_bloc: ^3.0.0
  counter_bloc:
    path: ../counter_bloc

flutter:
  uses-material-design: true
```

package:flutter_bloc

```
name: counter_app
description: A counter app example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_bloc: ^3.0.0
  counter_bloc:
    path: ../counter_bloc

flutter:
  uses-material-design: true
```

flutter_bloc in Action

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

void main() => runApp(CounterApp());
```

flutter_bloc in Action

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

void main() => runApp(CounterApp());
```

flutter_bloc in Action

```
...

class CounterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: BlocProvider(
        create: (_) => CounterBloc(),
        child: CounterPage(),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: BlocProvider(
        create: (_) => CounterBloc(),
        child: CounterPage(),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: BlocProvider(
        create: (_) => CounterBloc(),
        child: CounterPage(),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: BlocProvider(
        create: (_) => CounterBloc(),
        child: CounterPage(),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: BlocProvider(
        create: (_) => CounterBloc(),
        child: CounterPage(),
      ),
    );
  }
}
```

Bloc**Provider** 🤔

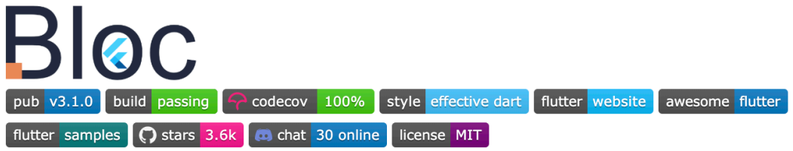
BlocProvider 🧐

flutter_bloc 3.1.0

Published Jan 1, 2020 • bloclibrary.dev 160 likes

FLUTTER ANDROID IOS WEB

Readme Changelog Example Installing Versions 100



A Flutter package that helps implement the BLoC pattern.

This package is built to work with bloc.

Bloc Widgets

BlocBuilder is a Flutter widget which requires a `Bloc` and a `builder` function. `BlocBuilder` handles building the widget in response to new states. `BlocBuilder` is very similar to `StreamBuilder` but has a more simple API to reduce the amount of boilerplate code needed. The `builder` function will potentially be called many times and should be a pure function that returns a widget in response to the state.

See `BlocListener` if you want to "do" anything in response to state changes such as navigation, showing a dialog, etc...

Publisher
bloclibrary.dev

About
Flutter Widgets that make it easy to implement the BLoC (Business Logic Component) design pattern. Built to be used with the bloc state management package.
Homepage
Repository (GitHub)
View/report issues
API reference

License
MIT (LICENSE)

Dependencies
bloc, flutter, provider

More
Packages that depend on flutter_bloc

Bloc**Provider** 😊

Dependencies

bloc, flutter, provider

https://imgflip.com/memetemplate/213059118/Surprised-Baby-Yoda

BlocProvider Anatomy

makes a bloc available to a sub-tree

```
BlocProvider(
  create: (BuildContext context) {
    return MyBloc();
  },
  child: TheChild(),
)
```

BlocProvider Anatomy

makes a bloc available to a sub-tree

```
BlocProvider(
  create: (BuildContext context) {
    return MyBloc();
  },
  child: TheChild(),
)
```

BlocProvider Anatomy

makes a bloc available to a sub-tree

```
BlocProvider(
  create: (BuildContext context) {
    return MyBloc();
  },
  child: TheChild(),
)
```

https://www.indiewire.com/2019/12/the-mandalorian-merchandise-questions-1202195206/

BlocProvider Anatomy

```
class TheChild extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final myBloc = BlocProvider.of<MyBloc>(context);

    ...
  }
}
```

BlocProvider Anatomy

```
class TheChild extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final myBloc = BlocProvider.of<MyBloc>(context);

    ...
  }
}
```

dispose?

```
...

class _CounterAppState extends State<CounterApp> {
  CounterBloc _counterBloc;

  @override
  void initState() {
    super.initState();
    _counterBloc = CounterBloc();
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    _counterBloc.close();
    super.dispose();
  }
}
```

dispose?

```
...

class _CounterAppState extends State<CounterApp> {
  CounterBloc _counterBloc;

  @override
  void initState() {
    super.initState();
    _counterBloc = CounterBloc();
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    _counterBloc.close();
    super.dispose();
  }
}
```

dispose?

```
...

class _CounterAppState extends State<CounterApp> {
  CounterBloc _counterBloc;

  @override
  void initState() {
    super.initState();
    _counterBloc = CounterBloc();
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    _counterBloc.close();
    super.dispose();
  }
}
```

BlocProvider Anatomy

by default BlocProvider automatically closes the provided bloc

```
BlocProvider({
    Key key,
    @required Create<T> create,
    Widget child,
    bool lazy,
  }) : this._(
          key: key,
          create: create,
          dispose: (_, bloc) => bloc?.close(),
          child: child,
          lazy: lazy,
        );
```

Multiple Blocs?

```
MultiBlocProvider(
  providers: [
    BlocProvider<BlocA>(
      create: (_) => BlocA(),
    ),
    BlocProvider<BlocB>(
      create: (_) => BlocB(),
    ),
  ],
  child: MyChild(),
)
```

Multiple Blocs?

```
MultiBlocProvider(
  providers: [
    BlocProvider<BlocA>(
      create: (_) => BlocA(),
    ),
    BlocProvider<BlocB>(
      create: (_) => BlocB(),
    ),
  ],
  child: MyChild(),
)
```

Multiple Blocs?

```
MultiBlocProvider(
  providers: [
    BlocProvider<BlocA>(
      create: (_) => BlocA(),
    ),
    BlocProvider<BlocB>(
      create: (_) => BlocB(),
    ),
  ],
  child: MyChild(),
)
```

Back to flutter_bloc in Action

```
...

class CounterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: BlocProvider(
        create: (_) => CounterBloc(),
        child: CounterPage(),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

BlocBuilder Anatomy

handles building a widget in response to bloc states

```
BlocBuilder<MyBloc, MyState>(
  builder: (BuildContext context, MyState state) {
    // return widget based on MyState
  }
)
```

BlocBuilder Anatomy

handles building a widget in response to bloc states

```
BlocBuilder<MyBloc, MyState>(
  builder: (BuildContext context, MyState state) {
    // return widget based on MyState
  }
)
```

BlocBuilder Anatomy

handles building a widget in response to bloc states

```
BlocBuilder<MyBloc, MyState>(
  builder: (BuildContext context, MyState state) {
    // return widget based on MyState
  }
)
```

Where is the Bloc?

```
BlocBuilder<MyBloc, MyState>(
  bloc: BlocProvider.of<MyBloc>(context),
  builder: (BuildContext context, MyState state) {
    // return widget based on MyState
  }
)
```

Where is the Bloc?

```
class _BlocBuilderBaseState<B extends Bloc<dynamic, S>, S>
    extends State<BlocBuilderBase<B, S>> {
  ...

  @override
  void initState() {
    super.initState();
    _bloc = widget.bloc ?? BlocProvider.of<B>(context);

    ...
  }
```

Back to flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

Back to flutter_bloc in Action

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocBuilder<CounterBloc, int>(
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
    );
  }
}
```

flutter_bloc in Action (Async)

```
...

BlocBuilder<CounterBloc, int>(
  builder: (context, state) {
    if (state is CounterLoadInProgress) {
      return MyLoadingIndicator();
    }
    if (state is CounterLoadSuccess) {
      return Text('You have pushed the button ${state.count} times.');
    }
    return MyError();
  },
),
```

flutter_bloc in Action (Async)

```
...

BlocBuilder<CounterBloc, int>(
  builder: (context, state) {
    if (state is CounterLoadInProgress) {
      return MyLoadingIndicator();
    }
    if (state is CounterLoadSuccess) {
      return Text('You have pushed the button ${state.count} times.');
    }
    return MyError();
  },
),
```

flutter_bloc in Action (Async)

```
...

BlocBuilder<CounterBloc, int>(
  builder: (context, state) {
    if (state is CounterLoadInProgress) {
      return MyLoadingIndicator();
    }
    if (state is CounterLoadSuccess) {
      return Text('You have pushed the button ${state.count} times.');
    }
    return MyError();
  },
),
```

flutter_bloc in Action (Async)

```
...

BlocBuilder<CounterBloc, int>(
  builder: (context, state) {
    if (state is CounterLoadInProgress) {
      return MyLoadingIndicator();
    }
    if (state is CounterLoadSuccess) {
      return Text('You have pushed the button ${state.count} times.');
    }
    return MyError();
  },
),
```

flutter_bloc in Action

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```

flutter_bloc in Action

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```

flutter_bloc in Action

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```

flutter_bloc in Action

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```

flutter_bloc in Action

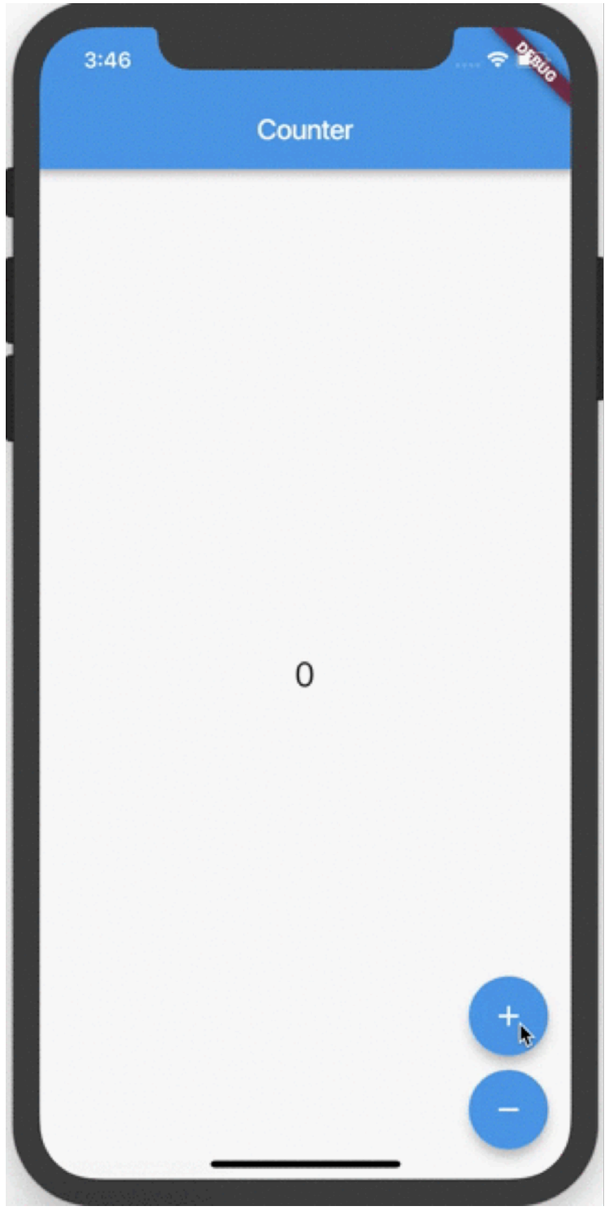
```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```

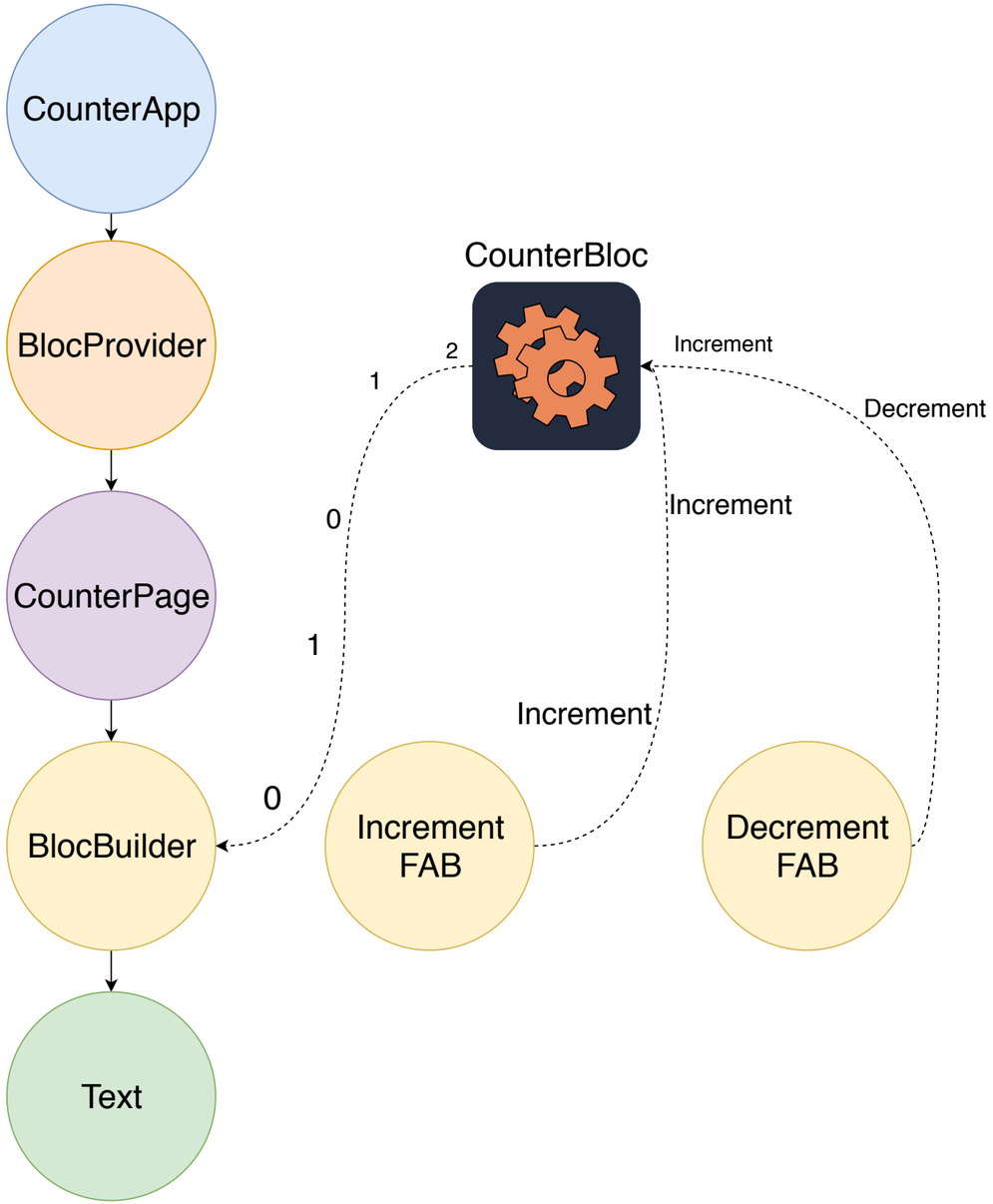
flutter_bloc in Action

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```

flutter_bloc in Action

```
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      ...
      floatingActionButton: Column(
        children: <Widget>[
          FloatingActionButton(
            child: Icon(Icons.add),
            onPressed: () => counterBloc.add(CounterEvent.increment),
          ),
          FloatingActionButton(
            child: Icon(Icons.remove),
            onPressed: () => counterBloc.add(CounterEvent.decrement),
          ),
        ],
      ),
    );
  }
}
```


3:46
Counter
0


CounterApp
BlocProvider
CounterPage
BlocBuilder
Text
CounterBloc
Increment
Decrement
Increment
Increment
Increment FAB
Decrement FAB
2
1
0
1
0

Side Effects: Snack Bars

BlocListener Anatomy

handles doing "stuff" in response to state changes

```
BlocListener<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to state changes
  },
  child: TheChild(),
)
```

BlocListener Anatomy

handles doing "stuff" in response to state changes

```
BlocListener<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to state changes
  },
  child: TheChild(),
)
```

BlocListener Anatomy

handles doing "stuff" in response to state changes

```
BlocListener<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to state changes
  },
  child: TheChild(),
)
```

BlocListener Anatomy

handles doing "stuff" in response to state changes

```
BlocListener<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to state changes
  },
  child: TheChild(),
)
```

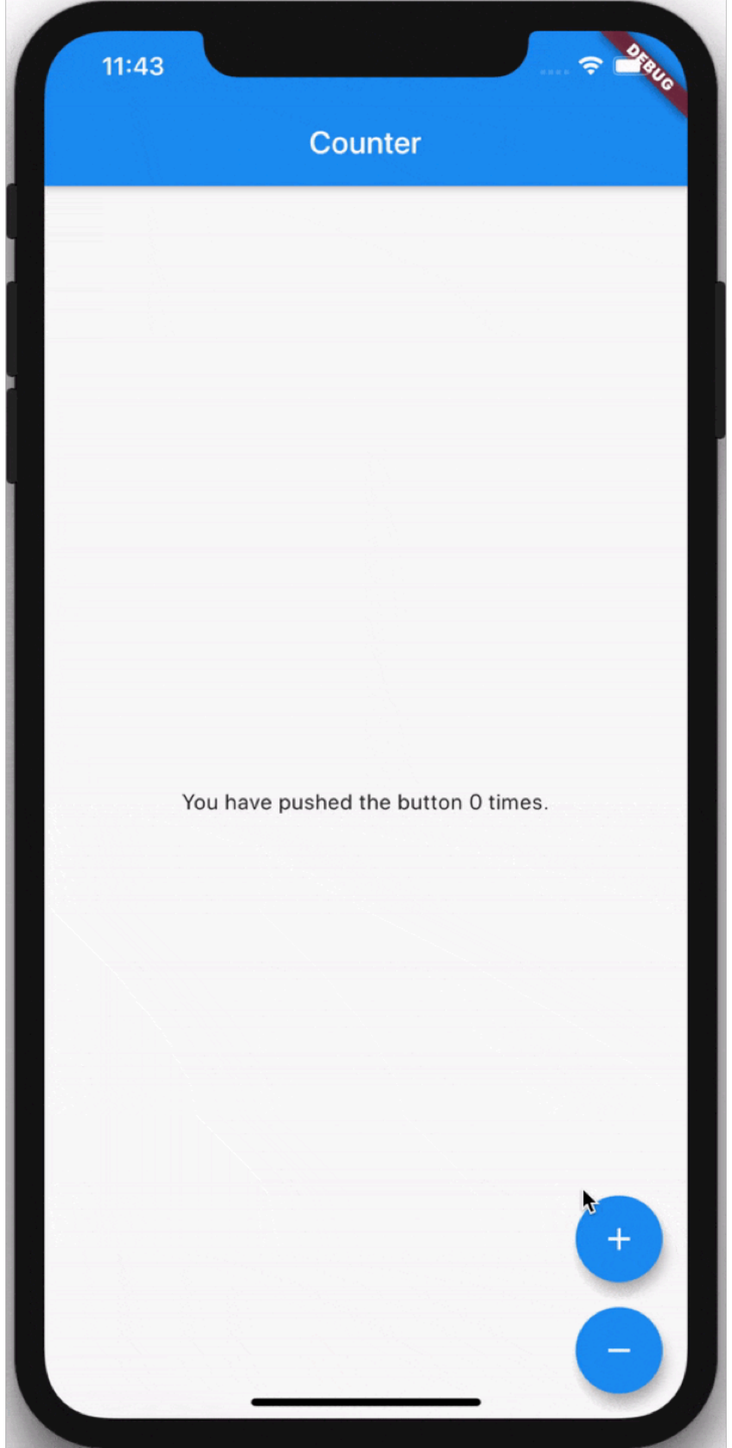
https://www.indiewire.com/2019/12/the-mandalorian-merchandise-questions-1202195206/

Back to Side Effects: Snack Bars

```
...
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocListener<CounterBloc, int>(
          listener: (context, state) {
            Scaffold.of(context)
              ..hideCurrentSnackBar()
              ..showSnackBar(SnackBar(content: Text('counter: $state')));
          },
          child: BlocBuilder<CounterBloc, int>(
            builder: (context, state) {
              return Text('You have pushed the button $state times.');
            },
          ),
        ),
      ),
      ...
```

Back to Side Effects: Snack Bars

```
...
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocListener<CounterBloc, int>(
          listener: (context, state) {
            Scaffold.of(context)
              ..hideCurrentSnackBar()
              ..showSnackBar(SnackBar(content: Text('counter: $state')));
          },
          child: BlocBuilder<CounterBloc, int>(
            builder: (context, state) {
              return Text('You have pushed the button $state times.');
            },
          ),
        ),
      ),
      ...
```


11:43
DEBUG
Counter
You have pushed the button 0 times.

https://makeameme.org/meme/brace-yourself-refactoring

BlocConsumer Anatomy

combined BlocBuilder and BlocListener

```
BlocConsumer<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to new states
  },
  builder: (BuildContext context, MyState state) {
    // return widgets in response to new states
  },
)
```

BlocConsumer Anatomy

combined BlocBuilder and BlocListener

```
BlocConsumer<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to new states
  },
  builder: (BuildContext context, MyState state) {
    // return widgets in response to new states
  },
)
```

BlocConsumer Anatomy

combined BlocBuilder and BlocListener

```
BlocConsumer<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to new states
  },
  builder: (BuildContext context, MyState state) {
    // return widgets in response to new states
  },
)
```

BlocConsumer Anatomy

combined BlocBuilder and BlocListener

```
BlocConsumer<MyBloc, MyState>(
  listener: (BuildContext context, MyState state) {
    // do stuff in response to new states
  },
  builder: (BuildContext context, MyState state) {
    // return widgets in response to new states
  },
)
```

The Refactor

```
...
class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocListener<CounterBloc, int>(
          listener: (context, state) {
            Scaffold.of(context)
              ..hideCurrentSnackBar()
              ..showSnackBar(SnackBar(content: Text('counter: $state')));
          },
          child: BlocBuilder<CounterBloc, int>(
            builder: (context, state) {
              return Text('You have pushed the button $state times.');
            },
          ),
        ),
      ),
      ...
```

The Refactor

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocConsumer<CounterBloc, int>(
          listener: (context, state) {
            Scaffold.of(context)
              ..hideCurrentSnackBar()
              ..showSnackBar(SnackBar(content: Text('counter: $state')));
          },
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
      ...
```

The Refactor

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocConsumer<CounterBloc, int>(
          listener: (context, state) {
            Scaffold.of(context)
              ..hideCurrentSnackBar()
              ..showSnackBar(SnackBar(content: Text('counter: $state')));
          },
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
      ...
```

The Refactor

```
...

class CounterPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterBloc = BlocProvider.of<CounterBloc>(context);
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(
        child: BlocConsumer<CounterBloc, int>(
          listener: (context, state) {
            Scaffold.of(context)
              ..hideCurrentSnackBar()
              ..showSnackBar(SnackBar(content: Text('counter: $state')));
          },
          builder: (context, state) {
            return Text('You have pushed the button $state times.');
          },
        ),
      ),
      ...
```

Goals

- ☑ decouple UI & business logic
- ☑ easy to test
- ☐ **predictable/maintainable**

bloc: onEvent

invoked when an event is added to the bloc

```
class CounterBloc extends Bloc<CounterEvent, int> {
  @override
  int get initialState => 0;

  @override
  void onEvent(CounterEvent event) {
    super.onEvent(event);
    print('onEvent $event');
  }

  ...
}
```

bloc: onEvent

invoked when an event is added to the bloc

```
class CounterBloc extends Bloc<CounterEvent, int> {
  @override
  int get initialState => 0;

  @override
  void onEvent(CounterEvent event) {
    super.onEvent(event);
    print('onEvent $event');
  }

  ...
}
```

the good old days

```
import 'counter_bloc.dart';

void main() {
    final counterBloc = CounterBloc();

    counterBloc.listen(print);

    counterBloc.add(CounterEvent.increment);
    counterBloc.add(CounterEvent.decrement);
}
```

onEvent in action

```
$ dart example/main.dart
onEvent CounterEvent.increment
onEvent CounterEvent.decrement
0
1
0
```

onEvent in action

```
$ dart example/main.dart
onEvent CounterEvent.increment
onEvent CounterEvent.decrement
0
1
0
```

onEvent in action

```
$ dart example/main.dart
onEvent CounterEvent.increment
onEvent CounterEvent.decrement
0
1
0
```

onEvent in action

```
$ dart example/main.dart
onEvent CounterEvent.increment
onEvent CounterEvent.decrement
0
1
0
```

bloc: onTransition

invoked when a new state is emitted

```
class CounterBloc extends Bloc<CounterEvent, int> {
  @override
  int get initialState => 0;

  @override
  void onTransition(Transition<CounterEvent, int> transition) {
    super.onTransition(transition);
    print('onTransition $transition');
  }

  ...
}
```

bloc: onTransition

invoked when a new state is emitted

```
class CounterBloc extends Bloc<CounterEvent, int> {
  @override
  int get initialState => 0;

  @override
  void onTransition(Transition<CounterEvent, int> transition) {
    super.onTransition(transition);
    print('onTransition $transition');
  }

  ...
}
```

onTransition in action

```
$ dart example/main.dart
0
onTransition Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
```

onTransition in action

```
$ dart example/main.dart
0
onTransition Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
```

onTransition in action

```
$ dart example/main.dart
0
onTransition Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
```

onTransition in action

```
$ dart example/main.dart
0
onTransition Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
```

onTransition in action

```
$ dart example/main.dart
0
onTransition Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
```

onTransition in action

```
$ dart example/main.dart
0
onTransition Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
```

bloc: onError

invoked when an uncaught error is thrown within a bloc

```
class CounterBloc extends Bloc<CounterEvent, int> {
  @override
  int get initialState => 0;

  @override
  void onError(Object error, StackTrace stacktrace) {
    super.onError(error, stacktrace);
    print('onError $error, $stacktrace');
  }

  ...
}
```

bloc: onError

invoked when an uncaught error is thrown within a bloc

```
class CounterBloc extends Bloc<CounterEvent, int> {
  @override
  int get initialState => 0;

  @override
  void onError(Object error, StackTrace stacktrace) {
    super.onError(error, stacktrace);
    print('onError $error, $stacktrace');
  }

  ...
}
```

bloc: onError

```
class CounterBloc extends Bloc<CounterEvent, int> {

  ...

  @override
  Stream<int> mapEventToState(CounterEvent event) async* {
    switch (event) {
      case CounterEvent.increment:
        yield state + 1;
        break;
      case CounterEvent.decrement:
        yield state - 1;
        break;
      default:
        throw Exception('unsupported event!');
    }
  }
}
```

bloc: onError

```
void main() {
  final counterBloc = CounterBloc();

  counterBloc.listen(print);

  counterBloc.add(CounterEvent.increment);
  counterBloc.add(CounterEvent.decrement);
  counterBloc.add(null);
}
```

onError in action

```
$ dart example/main.dart
0
1
0
onError Exception: unsupported event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:27:9) <asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

onError in action

```
$ dart example/main.dart
0
1
0
onError Exception: unsupported event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:27:9) <asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

onError in action

```
$ dart example/main.dart
0
1
0
onError Exception: unsupported event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:27:9) <asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

onError in action

```
$ dart example/main.dart
0
1
0
onError Exception: unsupported event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:27:9) <asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

seeing double: bloc delegate

handles hooks from all blocs

```dart
import 'package:bloc/bloc.dart';

class MyBlocDelegate extends BlocDelegate {}
```

seeing double: bloc delegate

```dart
import 'package:bloc/bloc.dart';

class MyBlocDelegate extends BlocDelegate {
  @override
  void onEvent(Bloc bloc, Object event) {
    super.onEvent(bloc, event);
    print('onEvent ${bloc.runtimeType}, $event');
  }

  @override
  void onTransition(Bloc bloc, Transition transition) {
    super.onTransition(bloc, transition);
    print('onTransition ${bloc.runtimeType}, $transition');
  }

  @override
  void onError(Bloc bloc, Object error, StackTrace stacktrace) {
    super.onError(bloc, error, stacktrace);
    print('onError ${bloc.runtimeType}, $error, $stacktrace');
  }
}
```

seeing double: bloc delegate

```dart
import 'package:bloc/bloc.dart';

class MyBlocDelegate extends BlocDelegate {
  @override
  void onEvent(Bloc bloc, Object event) {
    super.onEvent(bloc, event);
    print('onEvent ${bloc.runtimeType}, $event');
  }

  @override
  void onTransition(Bloc bloc, Transition transition) {
    super.onTransition(bloc, transition);
    print('onTransition ${bloc.runtimeType}, $transition');
  }

  @override
  void onError(Bloc bloc, Object error, StackTrace stacktrace) {
    super.onError(bloc, error, stacktrace);
    print('onError ${bloc.runtimeType}, $error, $stacktrace');
  }
}
```

seeing double: bloc delegate

```dart
import 'package:bloc/bloc.dart';

class MyBlocDelegate extends BlocDelegate {
  @override
  void onEvent(Bloc bloc, Object event) {
    super.onEvent(bloc, event);
    print('onEvent ${bloc.runtimeType}, $event');
  }

  @override
  void onTransition(Bloc bloc, Transition transition) {
    super.onTransition(bloc, transition);
    print('onTransition ${bloc.runtimeType}, $transition');
  }

  @override
  void onError(Bloc bloc, Object error, StackTrace stacktrace) {
    super.onError(bloc, error, stacktrace);
    print('onError ${bloc.runtimeType}, $error, $stacktrace');
  }
}
```

initialize bloc delegate

```dart
import 'package:bloc/bloc.dart';

import 'counter_bloc.dart';
import 'my_bloc_delegate.dart';

void main() {
  BlocSupervisor.delegate = MyBlocDelegate();
  final counterBloc = CounterBloc();

  counterBloc.listen(print);

  counterBloc.add(CounterEvent.increment);
  counterBloc.add(CounterEvent.decrement);
  counterBloc.add(null);
}
```

bloc delegate in action

```
$ dart example/main.dart
onEvent CounterBloc, CounterEvent.increment
onEvent CounterBloc, CounterEvent.decrement
onEvent CounterBloc, null
0
onTransition CounterBloc, Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition CounterBloc, Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
onError CounterBloc, Exception: unhandled event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:21:9)<asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

bloc delegate in action

```
$ dart example/main.dart
onEvent CounterBloc, CounterEvent.increment
onEvent CounterBloc, CounterEvent.decrement
onEvent CounterBloc, null
0
onTransition CounterBloc, Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition CounterBloc, Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
onError CounterBloc, Exception: unhandled event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:21:9)<asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

bloc delegate in action

```
$ dart example/main.dart
onEvent CounterBloc, CounterEvent.increment
onEvent CounterBloc, CounterEvent.decrement
onEvent CounterBloc, null
0
onTransition CounterBloc, Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition CounterBloc, Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
onError CounterBloc, Exception: unhandled event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:21:9)<asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

bloc delegate in action

```
$ dart example/main.dart
onEvent CounterBloc, CounterEvent.increment
onEvent CounterBloc, CounterEvent.decrement
onEvent CounterBloc, null
0
onTransition CounterBloc, Transition { currentState: 0, event: CounterEvent.increment, nextState: 1 }
1
onTransition CounterBloc, Transition { currentState: 1, event: CounterEvent.decrement, nextState: 0 }
0
onError CounterBloc, Exception: unhandled event!,
#0      CounterBloc.mapEventToState (file:///example/counter_bloc.dart:21:9)<asynchronous suspension>
#1      Bloc._bindStateSubject.<anonymous closure> (package:bloc/src/bloc.dart:155:14)
#2      Stream.asyncExpand.onListen.<anonymous closure> (dart:async/stream.dart:576:30)
#3      _RootZone.runUnaryGuarded (dart:async/zone.dart:1316:10)
#4      _BufferingStreamSubscription._sendData (dart:async/stream_impl.dart:338:11)
#5      _DelayedData.perform (dart:async/stream_impl.dart:593:14)
#6      _StreamImplEvents.handleNext (dart:async/stream_impl.dart:709:11)
#7      _PendingEvents.schedule.<anonymous closure> (dart:async/stream_impl.dart:669:7)
#8      _microtaskLoop (dart:async/schedule_microtask.dart:43:21)
#9      _startMicrotaskLoop (dart:async/schedule_microtask.dart:52:5)
#10     _runPendingImmediateCallback (dart:isolate-patch/isolate_patch.dart:118:13)
#11     _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:175:5)
```

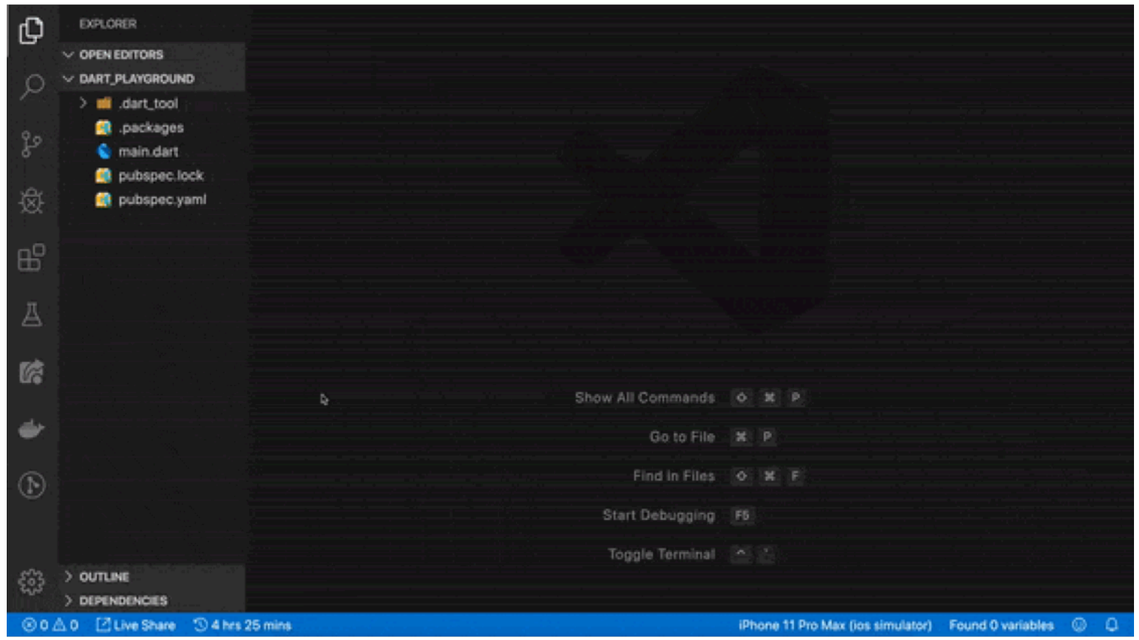
Goals

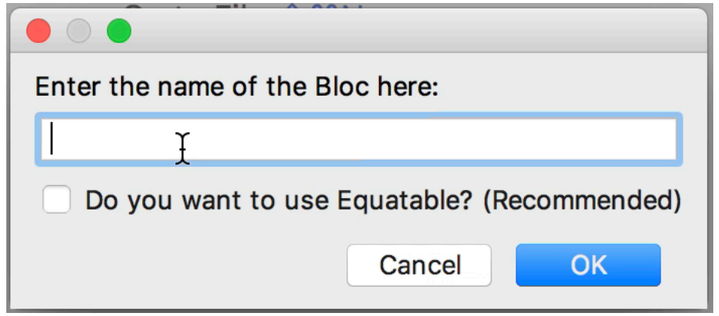
- ☑ decouple UI & business logic
- ☑ easy to test
- ☐ **predictable**/maintainable

Goals

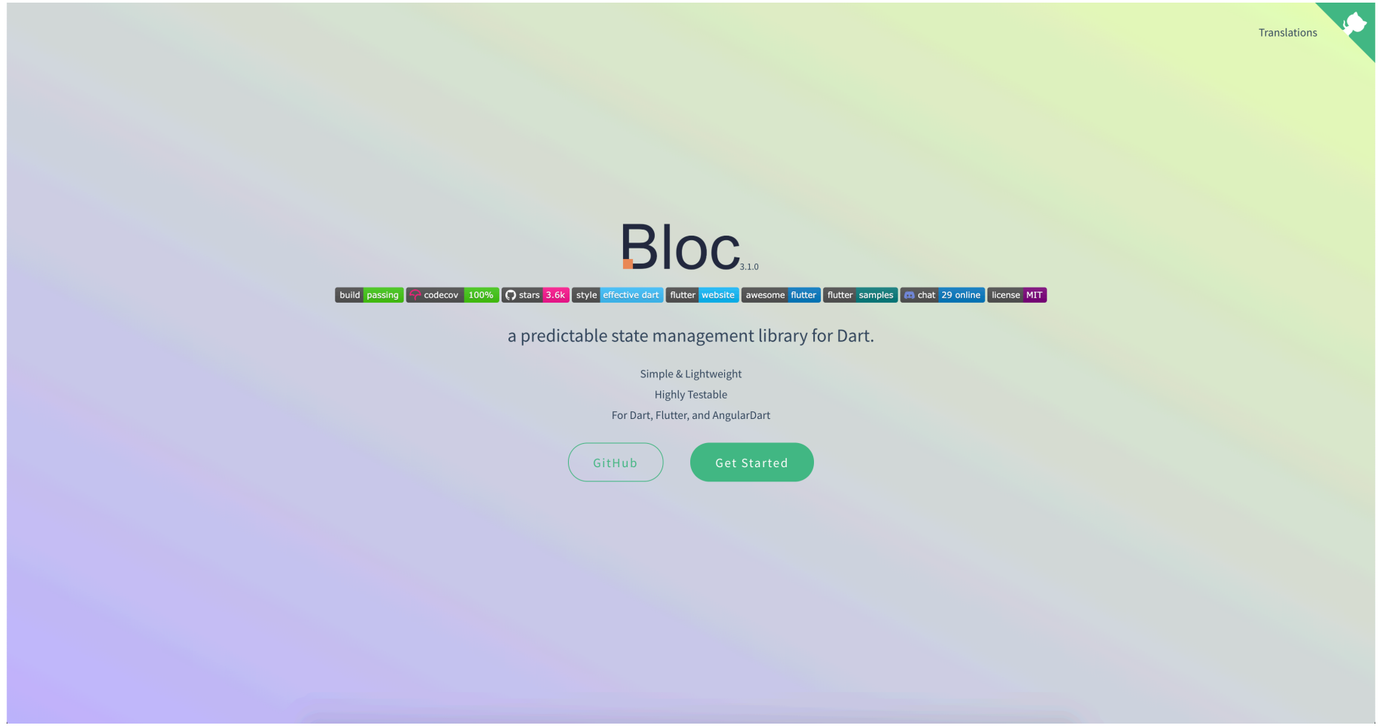
- ☑ decouple UI & business logic
- ☑ easy to test
- ☐ predictable/**maintainable**

Tooling


EXPLORER
OPEN EDITORS
DART_PLAYGROUND
.dart_tool
.packages
main.dart
pubspec.lock
pubspec.yaml
Show All Commands
Go to File
Find in Files
Start Debugging
Toggle Terminal
OUTLINE
DEPENDENCIES
Live Share
4 hrs 25 mins
iPhone 11 Pro Max (ios simulator)
Found 0 variables


Enter the name of the Bloc here:
Do you want to use Equatable? (Recommended)
Cancel
OK

Documentation



https://bloclibrary.dev

Examples Apps & Tutorials

Speaking of Examples...

Flutter Europe

LeanCode **Events**

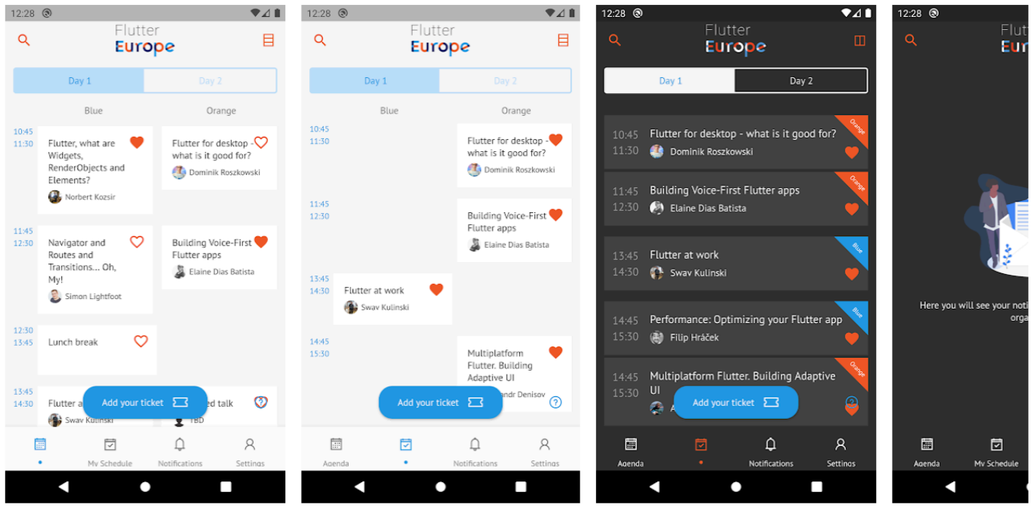
★★★★★ 15

PEGI 3

This app is compatible with all of your devices.

You can share this with your family. Learn more about Family Library

Installed



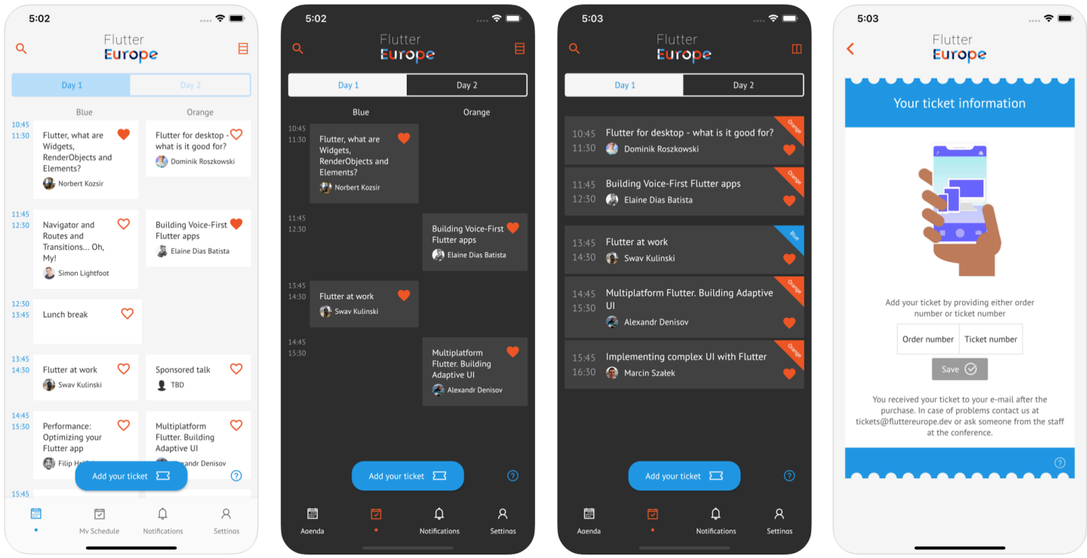
Flutter Europe 4+

Your app at Flutter Europe

LeanCode

Free

iPhone Screenshots



Goals

- ☑ decouple UI & business logic
- ☑ easy to test
- ☑ **predictable/maintainable**

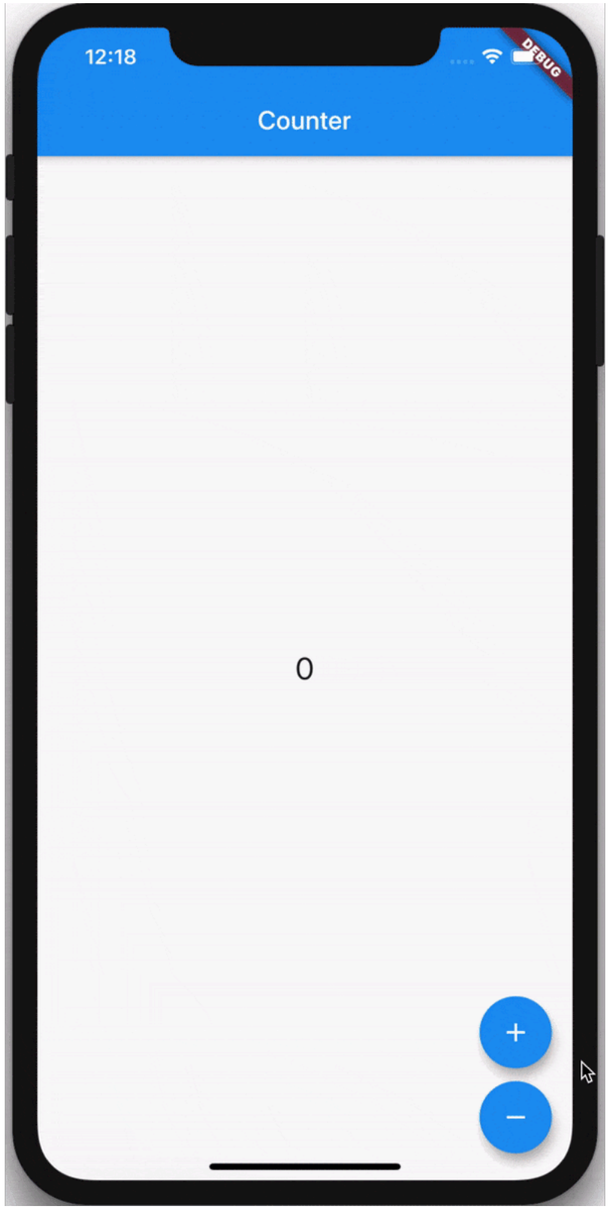
Bonus: package:hydrated_bloc

hydrated_bloc

96

An extension to the bloc state management library which automatically persists and restores bloc states.

v 3.0.0 • updated: Dec 25, 2019 • bloclibrary.dev FLUTTER


12:18
DEBUG
Counter
0
+
−

package:hydrated_bloc

```
name: counter_app
description: A counter app example
version: 1.0.0

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_bloc: ^3.0.0
  hydrated_bloc: ^3.0.0
  counter_bloc:
    path: ../counter_bloc

flutter:
  uses-material-design: true
```

Hydrated Bloc in Action

```
...

import 'package:hydrated_bloc/hydrated_bloc.dart';

class CounterBloc extends HydratedBloc<CounterEvent, int> {...}
```

Hydrated Bloc in Action

```
...

import 'package:hydrated_bloc/hydrated_bloc.dart';

class CounterBloc extends HydratedBloc<CounterEvent, int> {
  @override
  int get initialState => super.initialState ?? 0;
}
```

Hydrated Bloc in Action

```
...

import 'package:hydrated_bloc/hydrated_bloc.dart';

class CounterBloc extends HydratedBloc<CounterEvent, int> {
  @override
  int get initialState => super.initialState ?? 0;

  ...

  @override
  Map<String, int> toJson(int state) {
    return {'count': state};
  }

  @override
  int fromJson(Map<String, dynamic> source) {
    return source['count'] as int;
  }
}
```

Hydrated Bloc in Action

```
...

import 'package:hydrated_bloc/hydrated_bloc.dart';

class CounterBloc extends HydratedBloc<CounterEvent, int> {
  @override
  int get initialState => super.initialState ?? 0;

  ...

  @override
  Map<String, int> toJson(int state) {
    return {'count': state};
  }

  @override
  int fromJson(Map<String, dynamic> source) {
    return source['count'] as int;
  }
}
```

Hydrated Bloc in Action

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  BlocSupervisor.delegate = await HydratedBlocDelegate.build();
  runApp(App());
}
```

Hydrated Bloc in Action

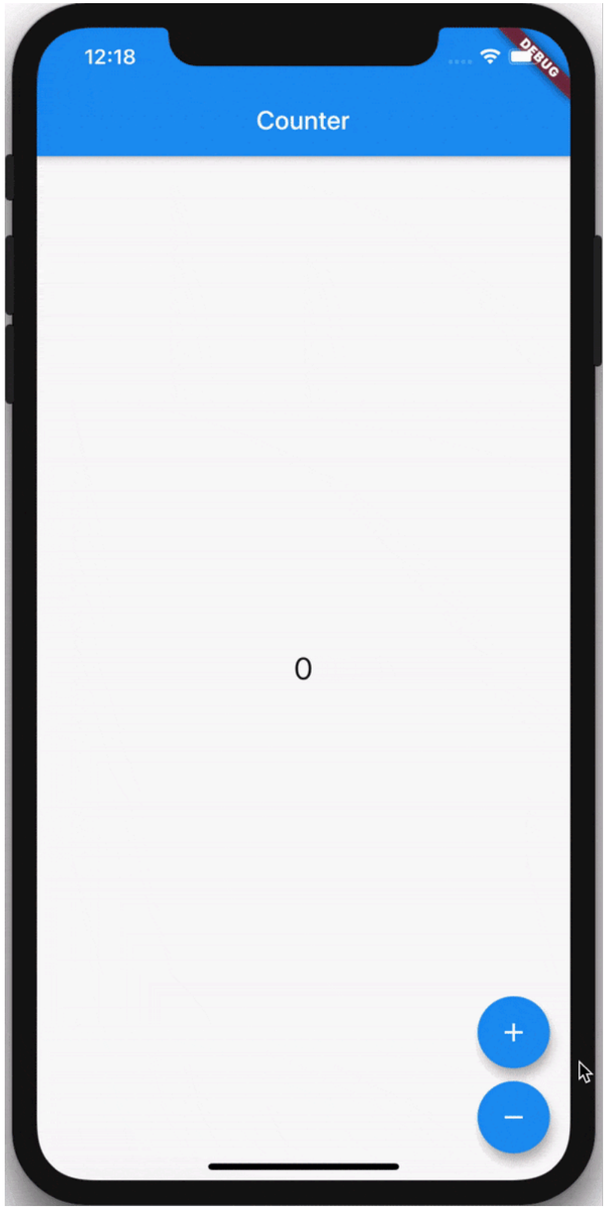
```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  BlocSupervisor.delegate = await HydratedBlocDelegate.build();
  runApp(App());
}
```

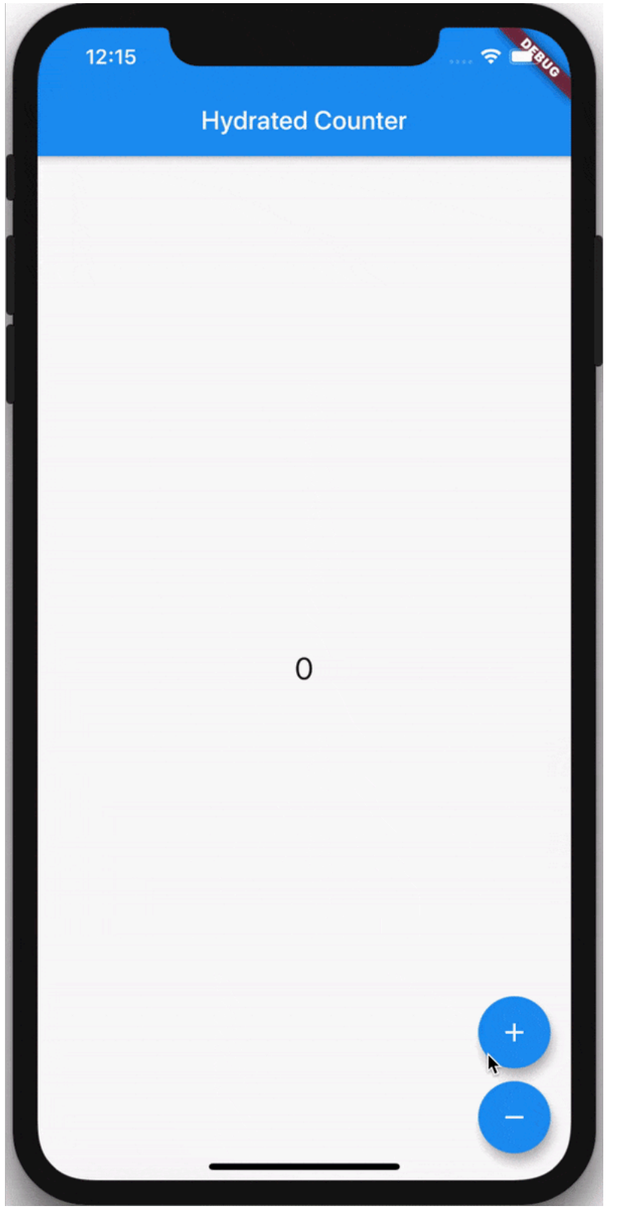
Hydrated Bloc in Action

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  BlocSupervisor.delegate = await HydratedBlocDelegate.build();
  runApp(App());
}
```

Hydrated Bloc in Action

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  BlocSupervisor.delegate = await HydratedBlocDelegate.build();
  runApp(App());
}
```


12:18
DEBUG
Counter
0


12:15
DEBUG
Hydrated Counter
0

Thanks for coming 👋

Twitter - @felangelov
Bloc Library Docs - https://bloclibrary.dev
Bloc Library Discord - https://discord.gg/Hc5KD3g

https://www.syfy.com/syfywire/baby-yoda-soup