

Cubit: Advanced Topics



Felix Angelov @ Very Good Ventures

Very Good Ventures, Chicago 😎🦄

Screenshot of a GitHub profile for Felix Angelov (felangel). The profile picture shows a smiling man with a beard. The bio reads:

Hi there! I'm a software engineer at Very Good Ventures in Chicago, IL. I'm currently working on the bloc library. I'm learning how to make latte art. I'm looking to collaborate on open source projects. Ask me about Flutter and Dart. Pronouns: he/him/his. Fun fact: I have a pet bunny named Coco.

The pinned repositories are:

- bloc: A predictable state management library that helps implement the BLoC design pattern. (Dart, 5.3k stars, 1.2k forks)
- bloc.js: A predictable state management library that helps implement the BLoC design pattern in JavaScript. (TypeScript, 67 stars, 8 forks)
- equatable: Simplify Equality Comparisons | A Dart abstract class that helps to implement equality without needing to explicitly override == and hashCode. (Dart, 377 stars, 37 forks)
- fresh: A token refresh library for Dart. (Dart, 92 stars, 7 forks)
- flutter: Forked from flutter/flutter. Flutter makes it easy and fast to build beautiful mobile apps. (Dart, 1 star)
- sealed_flutter_bloc: flutter_bloc state management extension that integrates sealed_unions. (Dart, 45 stars, 4 forks)

Follow, Sponsor, and other profile details are also visible.



<https://verygood.ventures>

Agenda

-  Testing with BlocTest
-  Caching with HydratedBloc
-  Undo/Redo with ReplayBloc

Testing with BlocTest

bloc_test 7.0.6

Published Oct 5, 2020 ·  [bloclibrary.dev](#)

DART NATIVE JS

FLUTTER ANDROID IOS WEB

80

[Readme](#)

[Changelog](#)

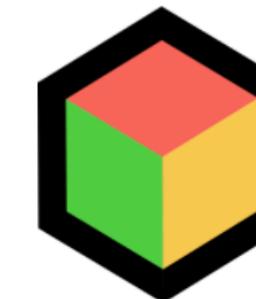
[Example](#)

[Installing](#)

[Versions](#)

[Scores](#)

[Admin](#)



bloc

pub v7.0.6





















Testing Goals



```
blocTest(  
  'emits correct weather',  
  build: () => WeatherCubit(),  
  act: (cubit) => cubit.fetchWeather(city: 'Warsaw'),  
  expect: [  
    WeatherState.loading(),  
    WeatherState.success(  
      Weather(  
        temperature: 10.3,  
        condition: WeatherCondition.cloudy,  
      ),  
    ),  
  ],  
)
```

Questions ?

- How can we control the emitted state?
- How can the test be consistent?
- How can we test error cases?

Answers



- How can we control the emitted state?
 - Decouple the cubit from the repository
- How can the test be consistent?
 - Inject a mock repository
- How can we test error cases?
 - Stub an exception

WeatherCubit

```
class WeatherCubit extends Cubit<WeatherState> {  
    ...  
  
    Future<void> fetchWeather({@required String city}) async {  
        emit(WeatherState.loading());  
        try {  
            final weather = await _weatherRepository.getWeather(city);  
            emit(WeatherState.success(weather));  
        } on Exception {  
            emit(WeatherState.failure());  
        }  
    }  
}
```

Handling Dependencies: The Easy Way

```
class WeatherCubit extends Cubit<WeatherState> {  
    WeatherCubit() : super(WeatherState.initial());  
  
    final WeatherRepository _weatherRepository = WeatherRepository();  
}
```

Handling Dependencies: The Better Way

```
class WeatherCubit extends Cubit<WeatherState> {  
    WeatherCubit(this._weatherRepository) : super(WeatherState.initial());  
  
    final WeatherRepository _weatherRepository;  
}
```

Let's take it from the top (of the widget tree)

```
void main() {  
    final weatherRepository = WeatherRepository();  
    runApp(  
        WeatherApp(  
            weatherRepository: weatherRepository,  
        ),  
    );  
}
```

Providing the Repository

```
class WeatherApp extends StatelessWidget {  
  WeatherApp({Key key, @required this.weatherRepository})  
    : super(key: key);  
  
  final WeatherRepository weatherRepository;  
  
  @override  
  Widget build(BuildContext context) {  
    return RepositoryProvider.value(  
      value: weatherRepository,  
      child: MaterialApp(...),  
    );  
  }  
}
```

What if there are many?

```
class WeatherApp extends StatelessWidget {  
    ...  
  
    @override  
    Widget build(BuildContext context) {  
        return MultiRepositoryProvider(  
            providers: [  
                RepositoryProvider.value(value: weatherRepository),  
                RepositoryProvider.value(value: userRepository),  
                ...  
            ],  
            child: MaterialApp(...),  
        );  
    }  
}
```

Providing the Cubit

```
class HomePage extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return BlocProvider(  
      create: (context) => WeatherCubit(  
        context.repository<WeatherRepository>(),  
      ),  
      child: HomeView(),  
    );  
  }  
}
```

Consuming the Cubit

```
class HomeView extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return BlocBuilder<WeatherCubit, WeatherState>(  
      builder: (context, state) {  
        switch (state.status) {  
          case WeatherStatus.initial:  
            return WeatherEmpty();  
          case WeatherStatus.loading:  
            return WeatherLoading();  
          case WeatherStatus.success:  
            return WeatherPopulated(weather: state.weather);  
          case WeatherStatus.failure:  
          default:  
            return WeatherError();  
        }  
      },  
    );  
  }  
}
```

Setting Up & Tearing Down

```
class MockWeatherRepository extends Mock implements WeatherRepository {}  
  
group('WeatherCubit', () {  
  WeatherRepository weatherRepository;  
  WeatherCubit weatherCubit;  
  
  setUp(() {  
    weatherRepository = MockWeatherRepository();  
    weatherCubit = WeatherCubit(weatherRepository);  
  });  
  
  tearDown(() {  
    weatherCubit.close();  
  });  
});
```

Stubbing the Repo

```
const warsawWeather = Weather(  
    temperature: 10.3,  
    condition: WeatherCondition.cloudy,  
);  
  
blocTest(  
    'emits weather when repository returns successfully',  
    build: () {  
        when(weatherRepository.getWeather(any))  
            .thenAnswer(_ async => warsawWeather);  
        return weatherCubit;  
    },  
    act: ...,  
    expect: ...  
);
```

Expectations

```
blocTest(  
  'emits weather when repository returns successfully',  
  build: ...  
  act: (cubit) => cubit.fetchWeather(city: 'Warsaw'),  
  expect: [  
    WeatherState.loading(),  
    WeatherState.success(warsawWeather),  
  ],  
);
```

Verification

```
blocTest(  
  'invokes getWeather with correct city name',  
  build: ...  
  act: (cubit) => cubit.fetchWeather(city: 'Warsaw'),  
  verify: (_) {  
    verify(weatherRepository.getWeather('Warsaw')).called(1);  
  },  
);
```

Rainy Day



```
const weatherException = Exception('weather-exception');

blocTest(
  'emits failure when repository throws',
  build: () {
    when(weatherRepository.getWeather(any)).thenThrow(weatherException);
    return weatherCubit;
  },
  act: (cubit) => cubit.fetchWeather(city: 'Warsaw'),
  expect: [
    WeatherState.loading(),
    WeatherState.failure(),
  ]
);
```

What about the Widgets? 🤔

Setting Up

```
class MockWeatherCubit extends MockBloc<WeatherState> implements WeatherCubit {}

group('HomeView', () {
  WeatherCubit weatherCubit;

  setUp(() {
    weatherCubit = MockWeatherCubit();
  });
});
```

Stubbing & Expecting the Initial State

```
testWidgets('renders WeatherEmpty when state is initial', (tester) async {
  when(weatherCubit.state).thenReturn(WeatherState.initial());
  await tester.pumpWidget(
    BlocProvider.value(
      value: weatherCubit,
      child: HomeView(),
    ),
  );
  expect(find.byType(WeatherEmpty), findsOneWidget);
});
```

What about the Loading State?

```
testWidgets('renders WeatherLoading when state is loading', (tester) async {
  when(weatherCubit.state).thenReturn(WeatherState.loading());
  await tester.pumpWidget(
    BlocProvider.value(
      value: weatherCubit,
      child: HomeView(),
    ),
  );
  expect(find.byType(WeatherLoading), findsOneWidget);
});
```

What about the Success State? ☀️

```
const warsawWeather = Weather(  
    temperature: 20.1,  
    condition: WeatherCondition.sunny,  
);  
  
testWidgets('renders WeatherPopulated when state is success', (tester) async {  
    when(weatherCubit.state).thenReturn(WeatherState.success(warsawWeather));  
    await tester.pumpWidget(  
        BlocProvider.value(  
            value: weatherCubit,  
            child: HomeView(),  
        ),  
    );  
    expect(find.byType(WeatherPopulated), findsOneWidget);  
});
```

What about the Failure State? 😱

```
testWidgets('renders WeatherError when state is failure', (tester) async {
  when(weatherCubit.state).thenReturn(WeatherState.failure());
  await tester.pumpWidget(
    BlocProvider.value(
      value: weatherCubit,
      child: HomeView(),
    ),
  );
  expect(find.byType(WeatherError), findsOneWidget);
});
```

Cubit Interactions

```
class SearchView extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      ...  
      floatingActionButton: FloatingActionButton(  
        child: Icon(Icons.search),  
        onPressed: () {  
          context.bloc<WeatherCubit>().fetchWeather('Warsaw');  
        }  
      ),  
    );  
  }  
}
```

Verifying Interactions

```
testWidgets('calls fetchWeather when search is pressed', (tester) async {
  when(weatherCubit.state).thenReturn(WeatherState.initial());
  await tester.pumpWidget(
    BlocProvider.value(
      value: weatherCubit,
      child: SearchView(),
    ),
  );
  await tester.tap(find.byType(FloatingActionButton));
  verify(weatherCubit.fetchWeather(city: 'Warsaw')).called(1);
});
```

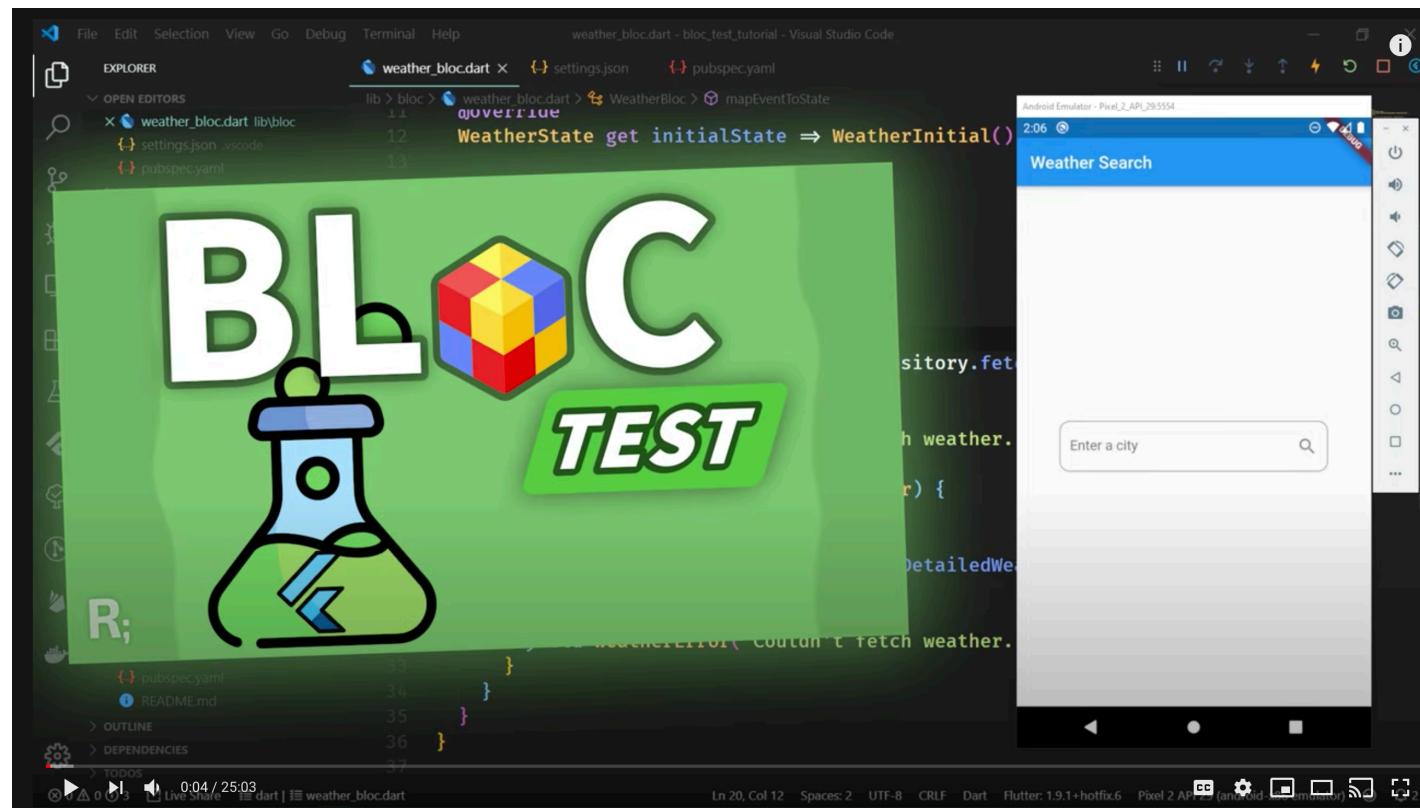
Side-Effects

```
class DetailsView extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return BlocListener<WeatherCubit, WeatherState>(  
      listener: (context, state) {  
        if (state.status == WeatherStatus.failure) {  
          Scaffold.of(context)  
            ..hideCurrentSnackBar()  
            ..showSnackBar(SnackBar(...));  
        }  
      },  
      child: ...  
    )  
  }  
}
```

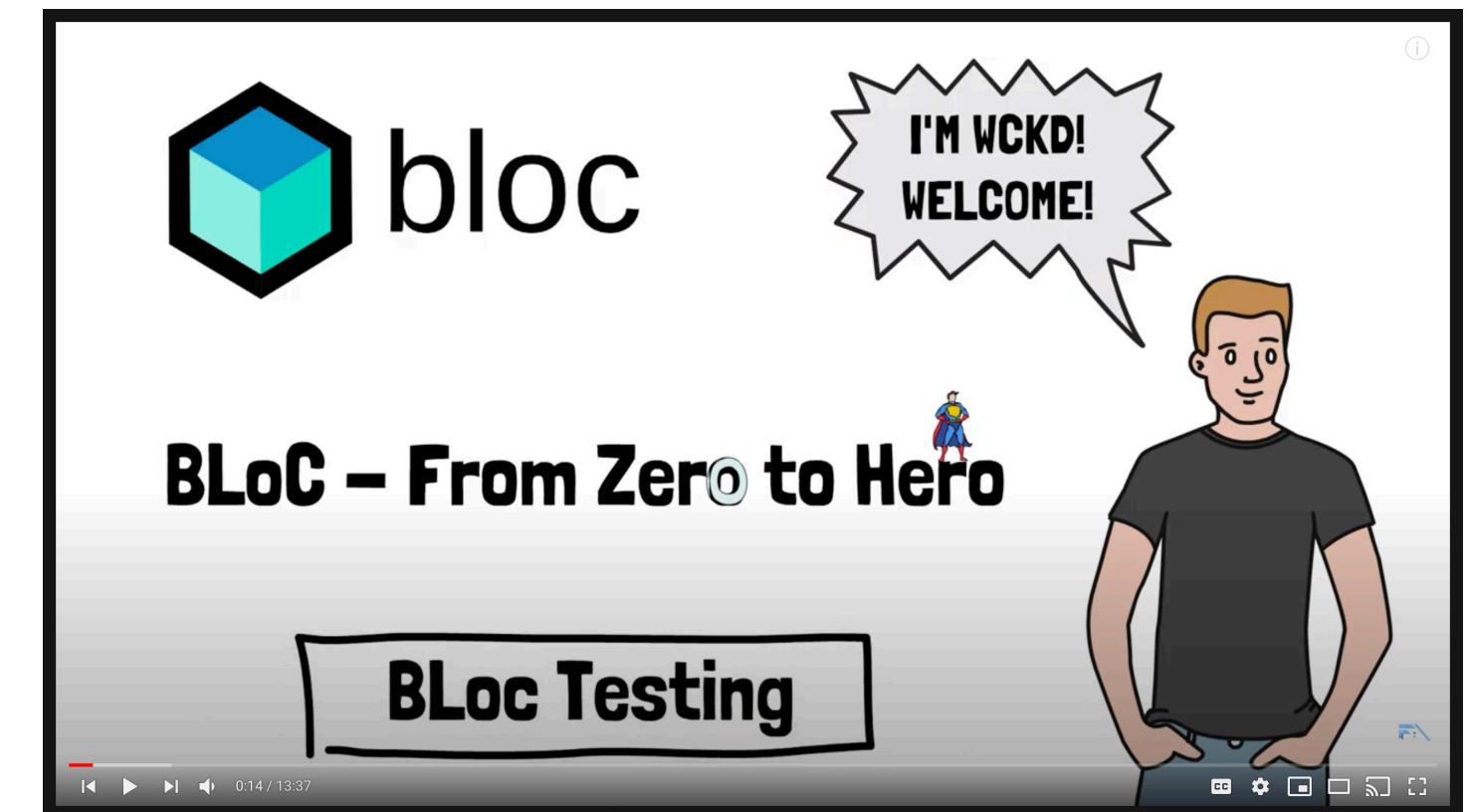
Testing Side-Effects (Listen Carefully 🧐)

```
testWidgets('shows Snackbar when state is failure', (tester) async {
  whenListen(weatherCubit, Stream.fromIterable([
    WeatherState.initial(),
    WeatherState.failure(),
  ]));
  await tester.pumpWidget(
    BlocProvider.value(
      value: weatherCubit,
      child: DetailsView(),
    ),
  );
  await tester.pump();
  expect(find.byTypeSnackBar, findsOneWidget);
});
```

More Testing Resources



<https://youtu.be/S6jFBiiP0Mc>



<https://youtu.be/cVru6Gy4duQ>

Caching with HydratedBloc

hydrated_bloc 6.0.1

Published Aug 5, 2020 ·  [bloclibrary.dev](#)

FLUTTER ANDROID IOS

 94

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#) [Admin](#)



How to Hydrate?



```
class CounterCubit extends Cubit<int> {  
    CounterCubit() : super(0);  
  
    void increment() => emit(state + 1);  
}
```

```
class CounterCubit extends HydratedCubit<int> {  
    CounterCubit() : super(0);  
  
    void increment() => emit(state + 1);  
  
    @override  
    int fromJson(Map<String, dynamic> json) {  
        return json['counter'] as int;  
    }  
  
    @override  
    Map<String, dynamic> toJson(int state) {  
        return {'counter': state};  
    }  
}
```

Undo/Redo with ReplayBloc

replay_bloc 0.0.1-dev.2

Published Aug 17, 2020 •  [bloclibrary.dev](#)

DART NATIVE JS

FLUTTER ANDROID IOS WEB

16

[Readme](#) Changelog Example Installing Versions Scores Admin



pub v0.0.1-dev.2

 build passing

 codecov 100%

 stars 5.6k

style effective dart

flutter website

awesome flutter

flutter samples

license MIT

 chat 109 online

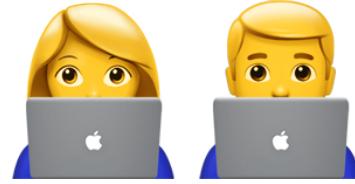
 bloc library

How to Replay?



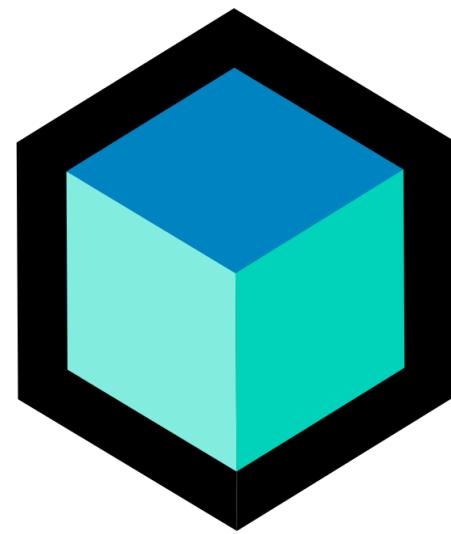
```
class CounterCubit extends Cubit<int> {  
    CounterCubit() : super(0);  
  
    void increment() => emit(state + 1);  
}
```

```
class CounterCubit extends ReplayCubit<int> {  
    CounterCubit() : super(0);  
  
    void increment() => emit(state + 1);  
}
```



Demo Time

https://github.com/felangel/cool_counter



bloc

Additional Resources

<https://bloclibrary.dev>

<https://discord.gg/Hc5KD3g>



Thank You!