Future<Flutter> 2024

Flutter Bloc을 제품 개발에 야무지게 적용하기

최정연, LINE / ABC Studio

Future<Flutter> 2024

https://github.com/hohoins/flutter_bloc_abc_studio

inflearn

새차원의 코틀린(Kotlin) 강좌

새차원

무료

4.5 (76) 8,400+

새차원의 코틀린 코루틴(Coroutines) 강좌

새차원

무료

4.9 (75) 2,700+

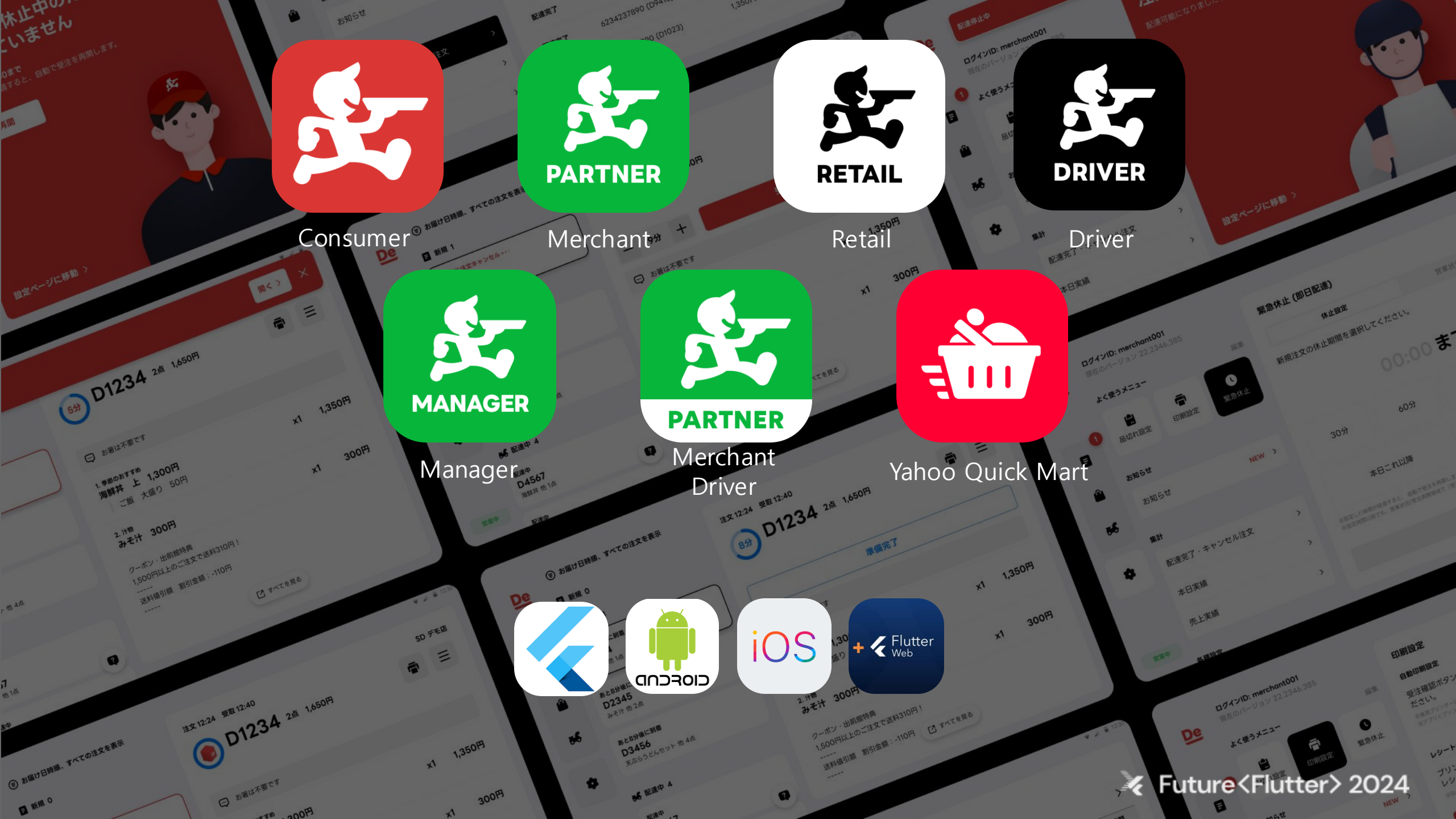
https://www.inflearn.com/courses?s=%EC%83%88%EC%B0%A8%EC%9B%90

LINE

ABC Studio

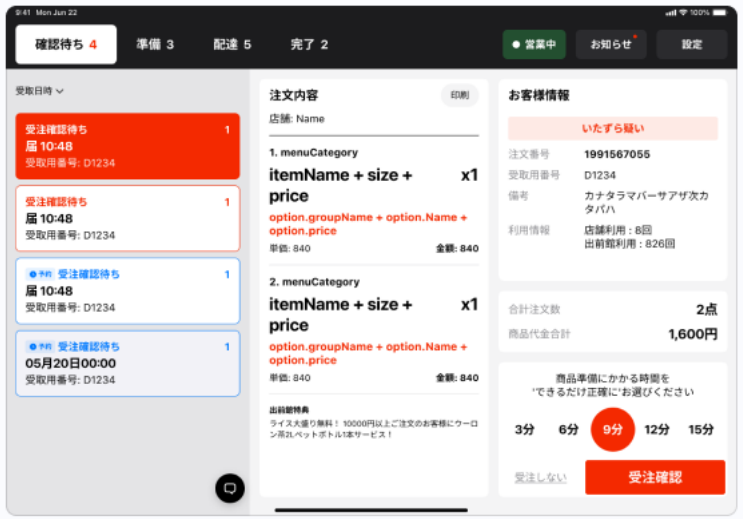
Flutter App 개발

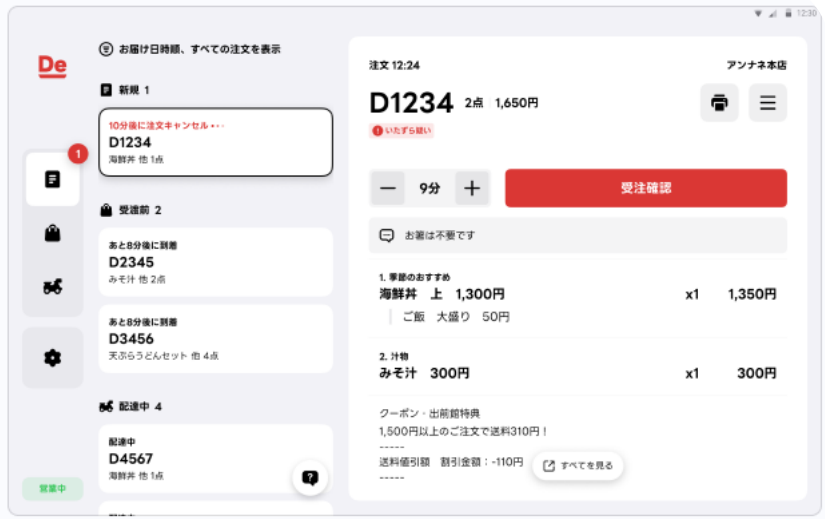
데마에칸 (일본 No.1 푸드 딜리버리)


PARTNER
RETAIL
DRIVER
Consumer
Merchant
Retail
Driver
MANAGER
PARTNER
Manager
Merchant
Driver
Yahoo Quick Mart
iOS
Flutter
Web
Future<Flutter> 2024

목차

1. Bloc을 선택한 이유





2021 2022 2023

리뉴얼 + 다양한 기능 추가 예정됨

프로덕트 규모 커짐

개발자 수 증가

보다 효과적 구조 고민

https://techblog.lycorp.co.jp/ko/how-to-ux-research-and-renewal-for-overseas-users

비즈니스 로직 분리 용이

높은 인지도 (유지보수⇧, 참고자료⇧)

bloc_test 용이

규모가 커진 앱에 적합하다고 판단

https://engineering.linecorp.com/ko/blog/flutter-architecture-getx-bloc-provider

2. 일회성 동작 처리

일회성 동작 ???

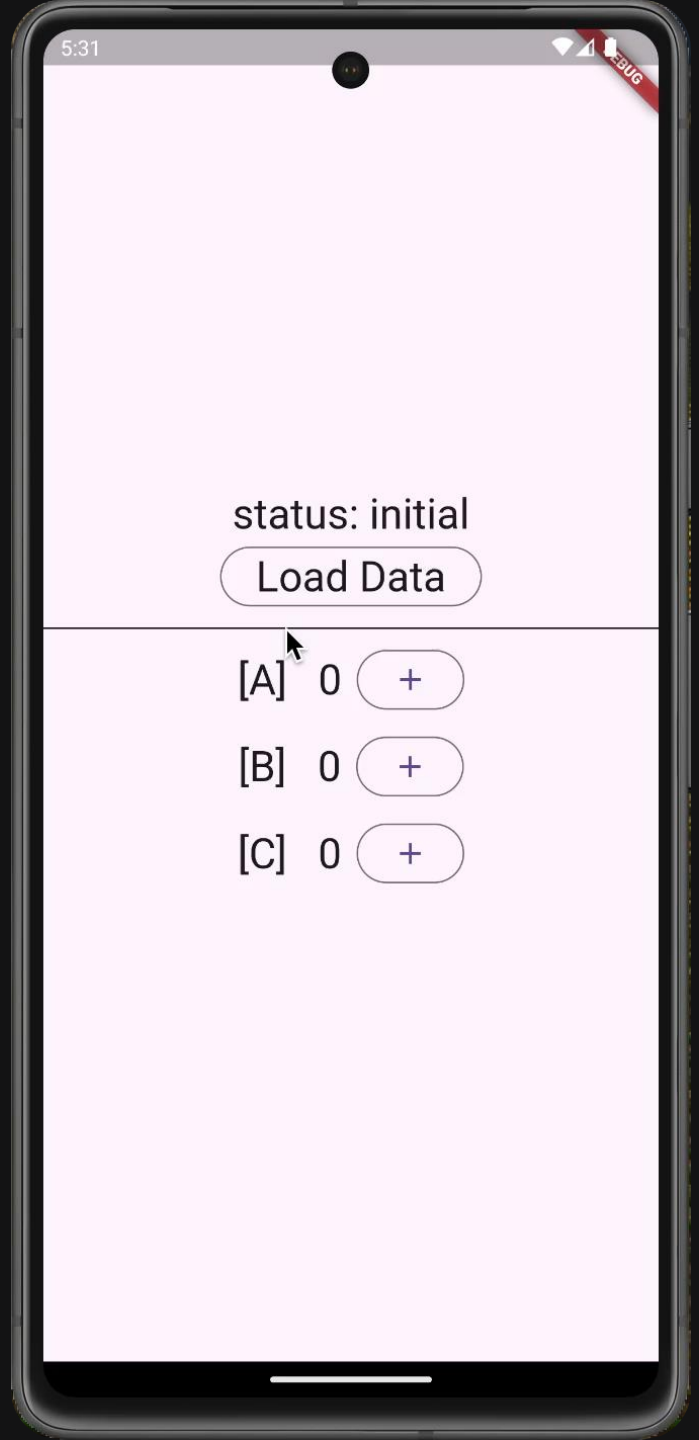
일회성 동작 ???

스낵바, 토스트 메시지, 다이얼로그

다음 화면으로 이동, 등등

스낵바

같이 만들어 봅시다


5:31
DEBUG
status: initial
Load Data
[A] 0 +
[B] 0 +
[C] 0 +

BlocListener

BlocListener는 필수 `BlocWidgetListener`와 선택적 `Bloc` 파라미터를 사용하고 bloc의 state 변경에 대한 응답으로 `listener`를 호출하는 Flutter 위젯입니다. Navigation, `Shackbar` 표시, `Dialog` 표시 등과 같이 state 변경당 한 번 발생해야 하는 기능에 사용해야 합니다.

`listener`는 `BlocBuilder`의 `builder`와 달리 각 state 변경 (초기 state를 포함하지 **않음**)에 대해 한 번만 호출되며 `void` 함수입니다.

```
BlocListener<BlocA, BlocAState>(
  listener: (context, state) {
    // do stuff here based on BlocA's state
  },
  child: const SizedBox(),
);
```

https://bloclibrary.dev/naming-conventions/

BlocListener

BlocListener는 필수 `BlocWidgetListener`와 선택적 `Bloc` 파라미터를 사용하고 bloc의 state 변경에 대한 응답으로 `listener`를 호출하는 Flutter 위젯입니다. Navigation, `Shackbar` 표시, `Dialog` 표시 등과 같이 state 변경당 한 번 발생해야 하는 기능에 사용해야 합니다.

`listener`는 `BlocBuilder`의 `builder`와 달리 각 state 변경 (초기 state를 포함하지 **않음**)에 대해 한 번만 호출되며 `void` 함수입니다.

```
BlocListener<BlocA, BlocAState>(
  listener: (context, state) {
    // do stuff here based on BlocA's state
  },
  child: const SizedBox(),
);
```

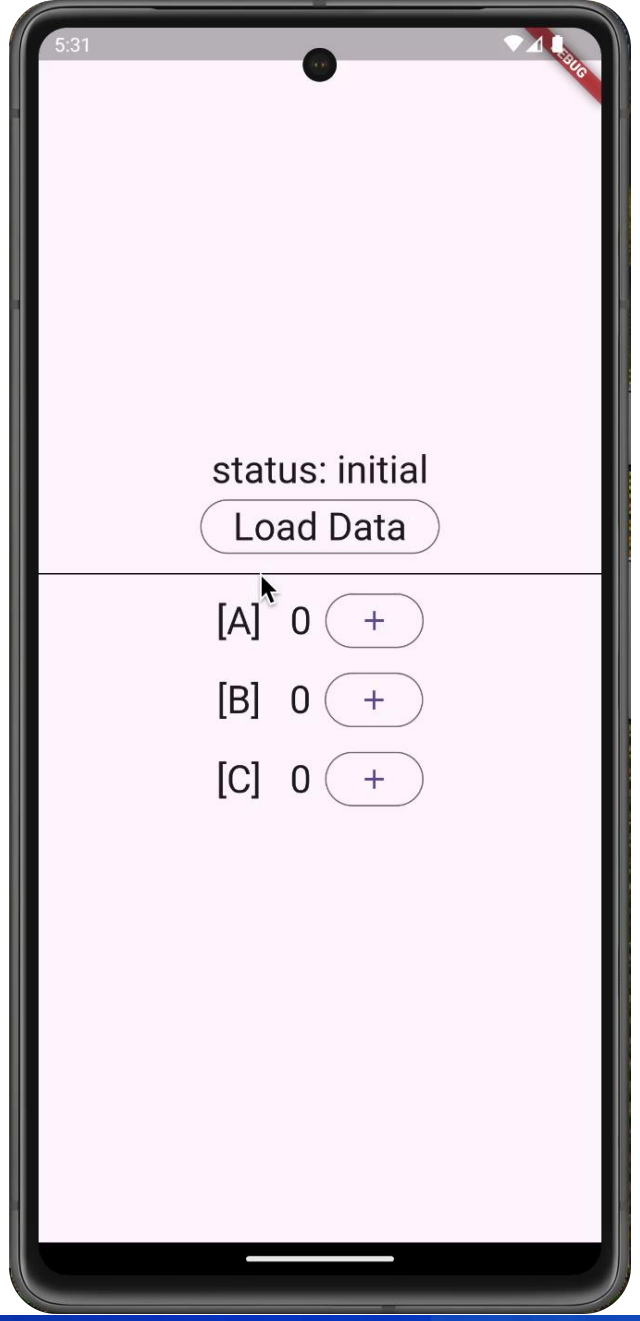
https://bloclibrary.dev/naming-conventions/

```dart
void onTapLoadData() async {
  emit(state.copyWith(status: OneTimeNgStatus.loading));
  await Future.delayed(const Duration(milliseconds: 500));
  emit(state.copyWith(status: OneTimeNgStatus.success));
}
```

```dart
void onTapLoadData() async {
  emit(state.copyWith(status: OneTimeNgStatus.loading));
  await Future.delayed(const Duration(milliseconds: 500));
  emit(state.copyWith(status: OneTimeNgStatus.success));
}
```

```dart
void onTapLoadData() async {
  emit(state.copyWith(status: OneTimeNgStatus.loading));
  await Future.delayed(const Duration(milliseconds: 500));
  emit(state.copyWith(status: OneTimeNgStatus.success));
}

return Scaffold(
  body: BlocListener<OneTimeNgCubit, OneTimeNgState>(
    listener: (context, state) {
      if (state.status == OneTimeNgStatus.success) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text('성공\n데이터 업데이트\nFuture<Flutter> 2024'),
            duration: Duration(milliseconds: 500),
          ),  // SnackBar
        );
      }
    },
    child: const _Contents(),
  ),  // BlocListener
);  // Scaffold
```

```dart
void onTapLoadData() async {
  emit(state.copyWith(status: OneTimeNgStatus.loading));
  await Future.delayed(const Duration(milliseconds: 500));
  emit(state.copyWith(status: OneTimeNgStatus.success));
}

return Scaffold(
  body: BlocListener<OneTimeNgCubit, OneTimeNgState>(
    listener: (context, state) {
      if (state.status == OneTimeNgStatus.success) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text('성공\n데이터 업데이트\nFuture<Flutter> 2024'),
            duration: Duration(milliseconds: 500),
          ),  // SnackBar
        );
      }
    },
    child: const _Contents(),
  ),  // BlocListener
);  // Scaffold
```

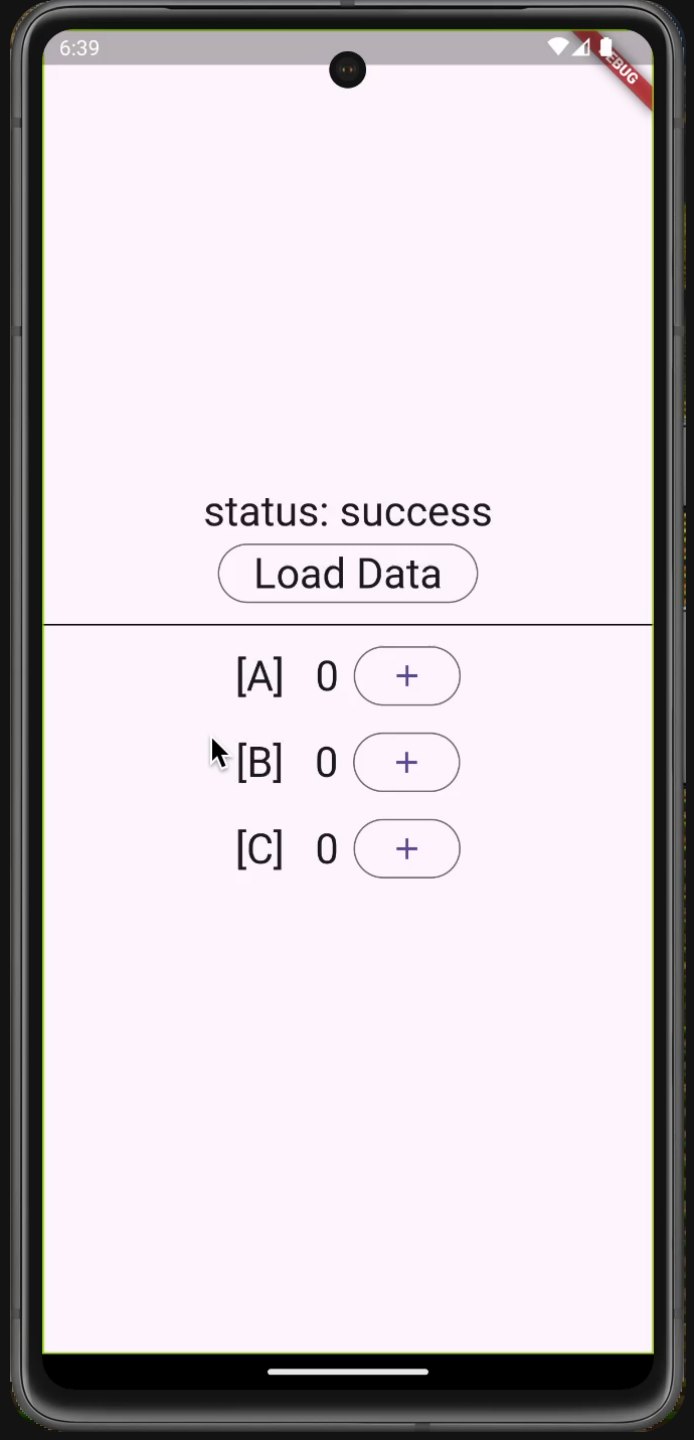


스낵바

간단하네요?

하지만...

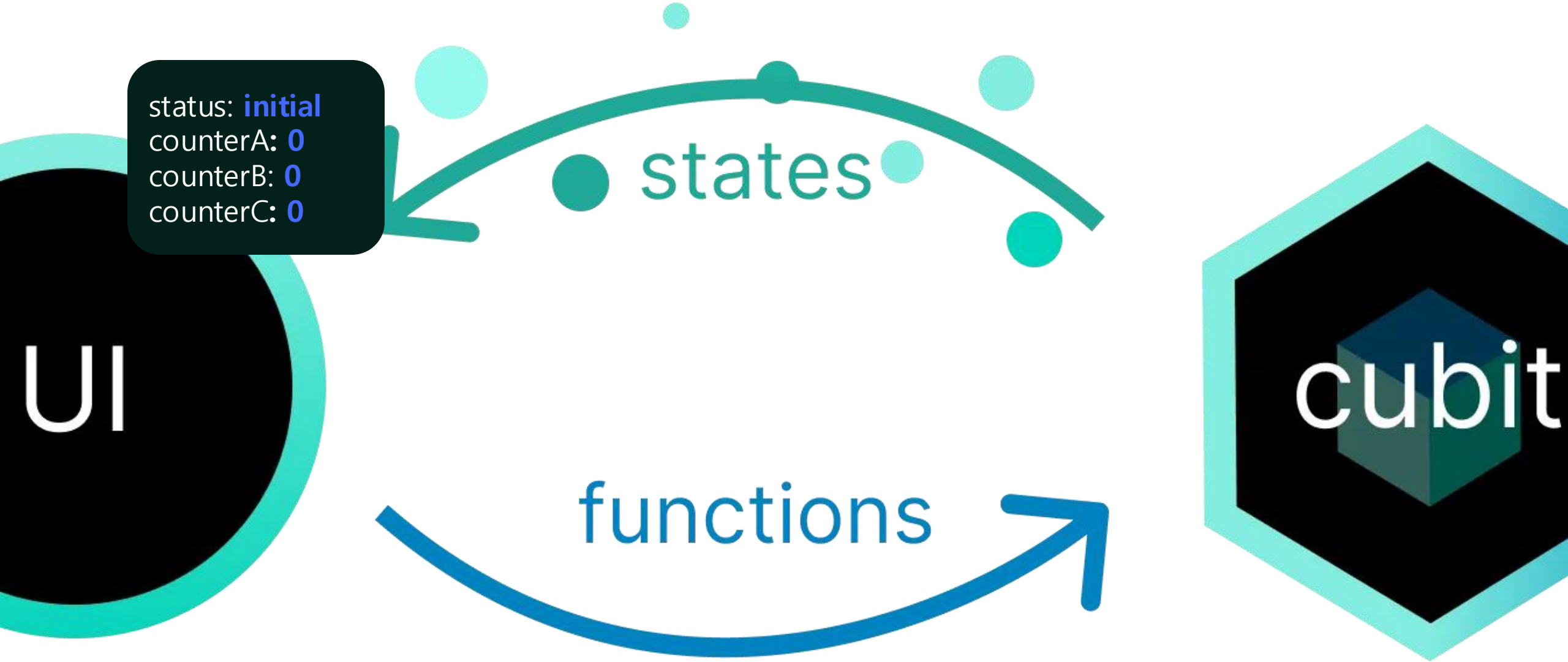
문제가 있습니다

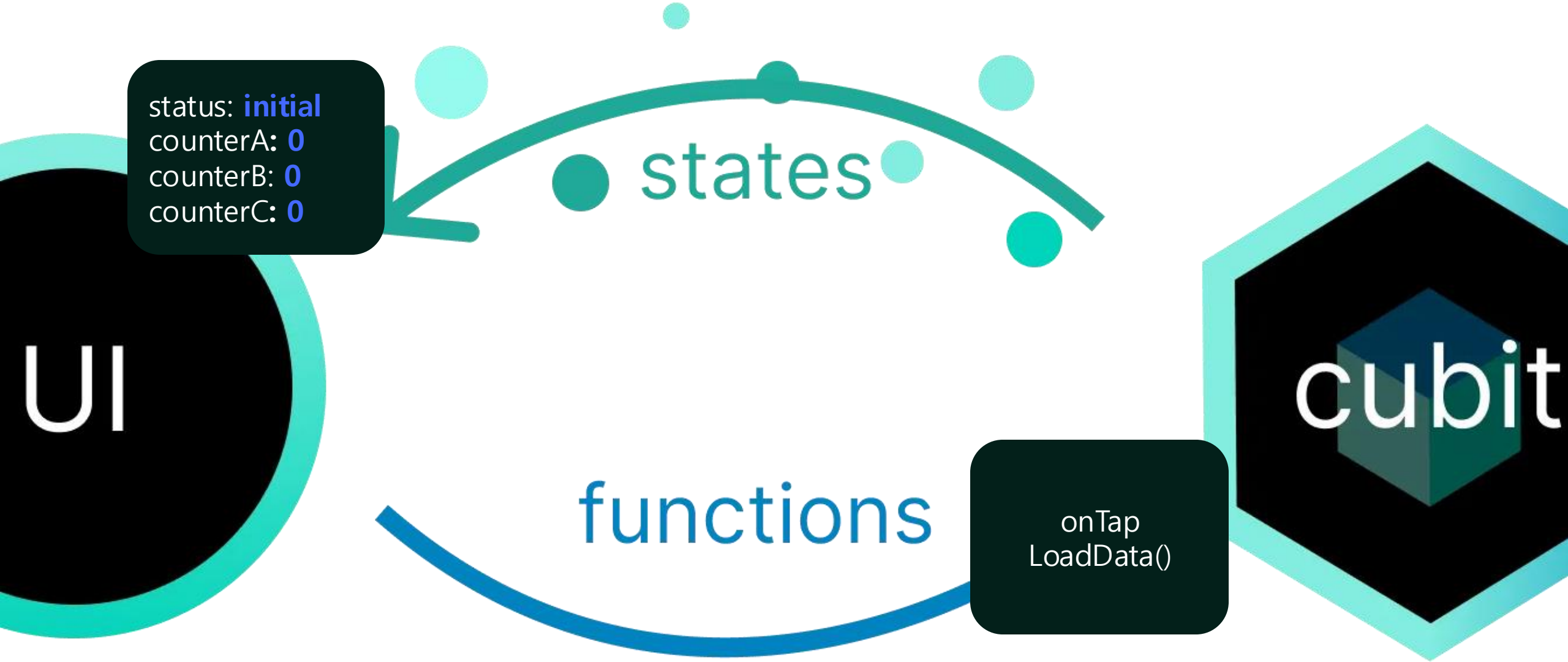

6:39
DEBUG
status: success
Load Data
[A] 0 +
[B] 0 +
[C] 0 +

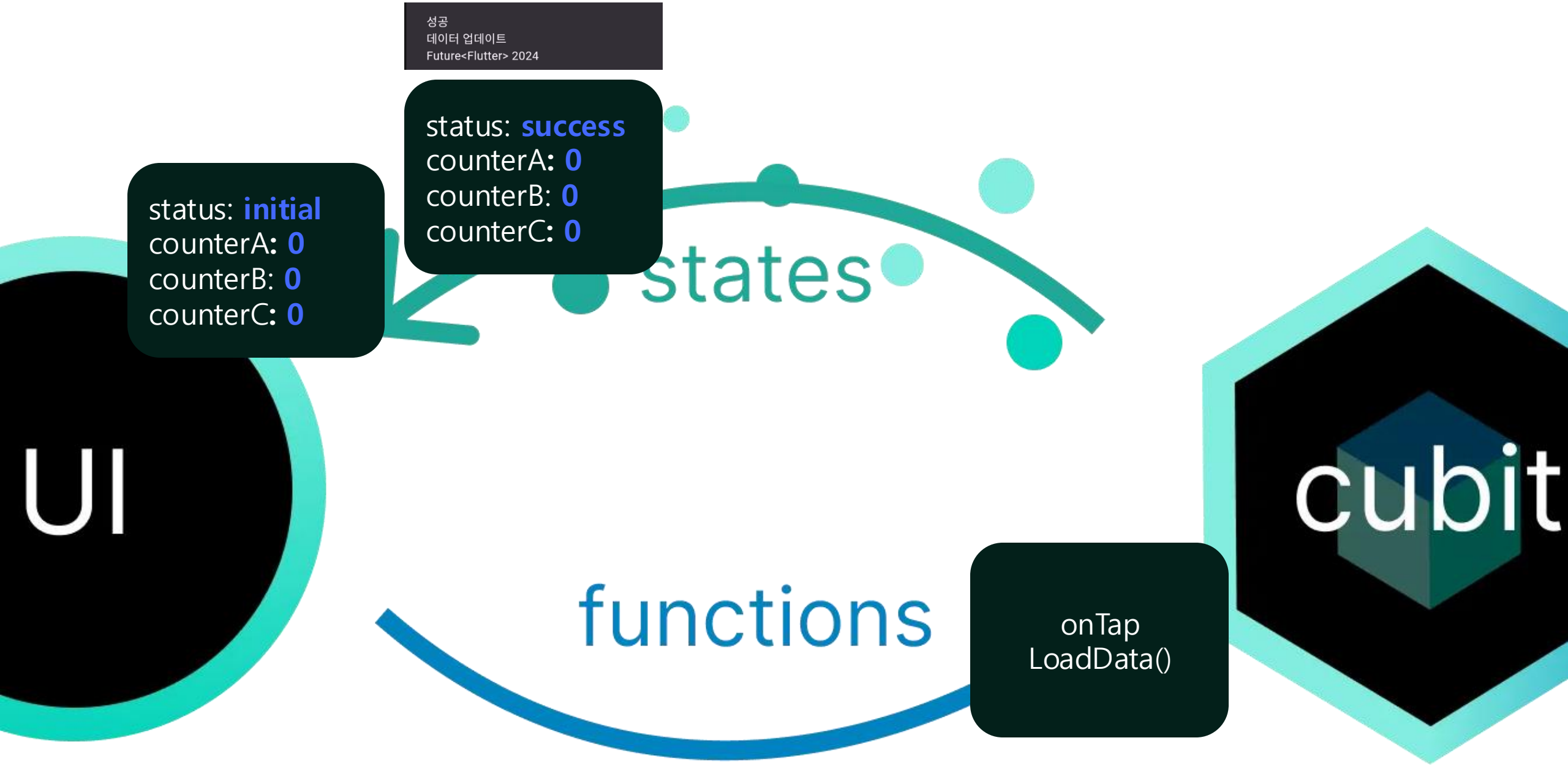
반복적 실행 문제 발생

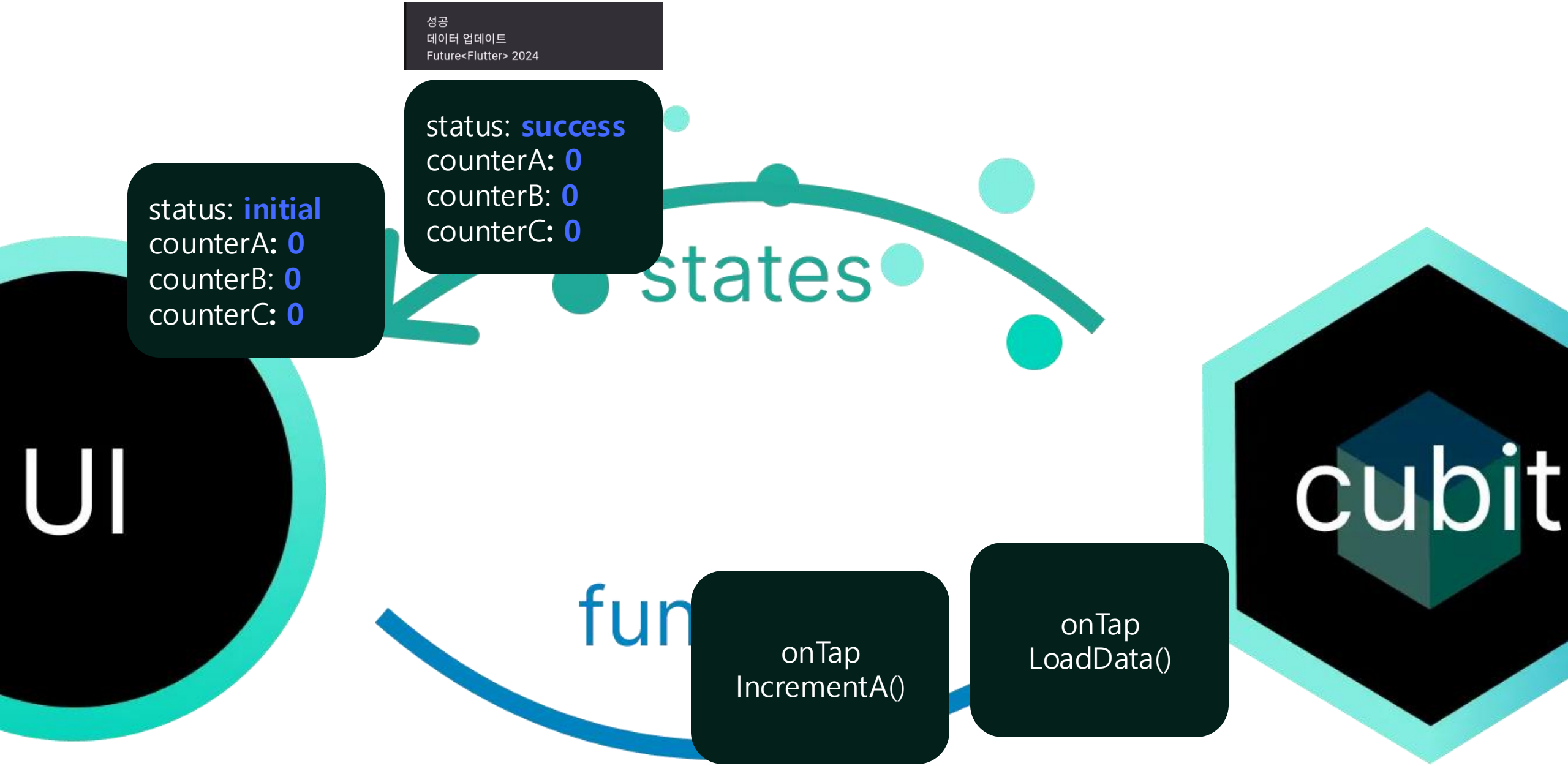
'+'를 눌렀는데

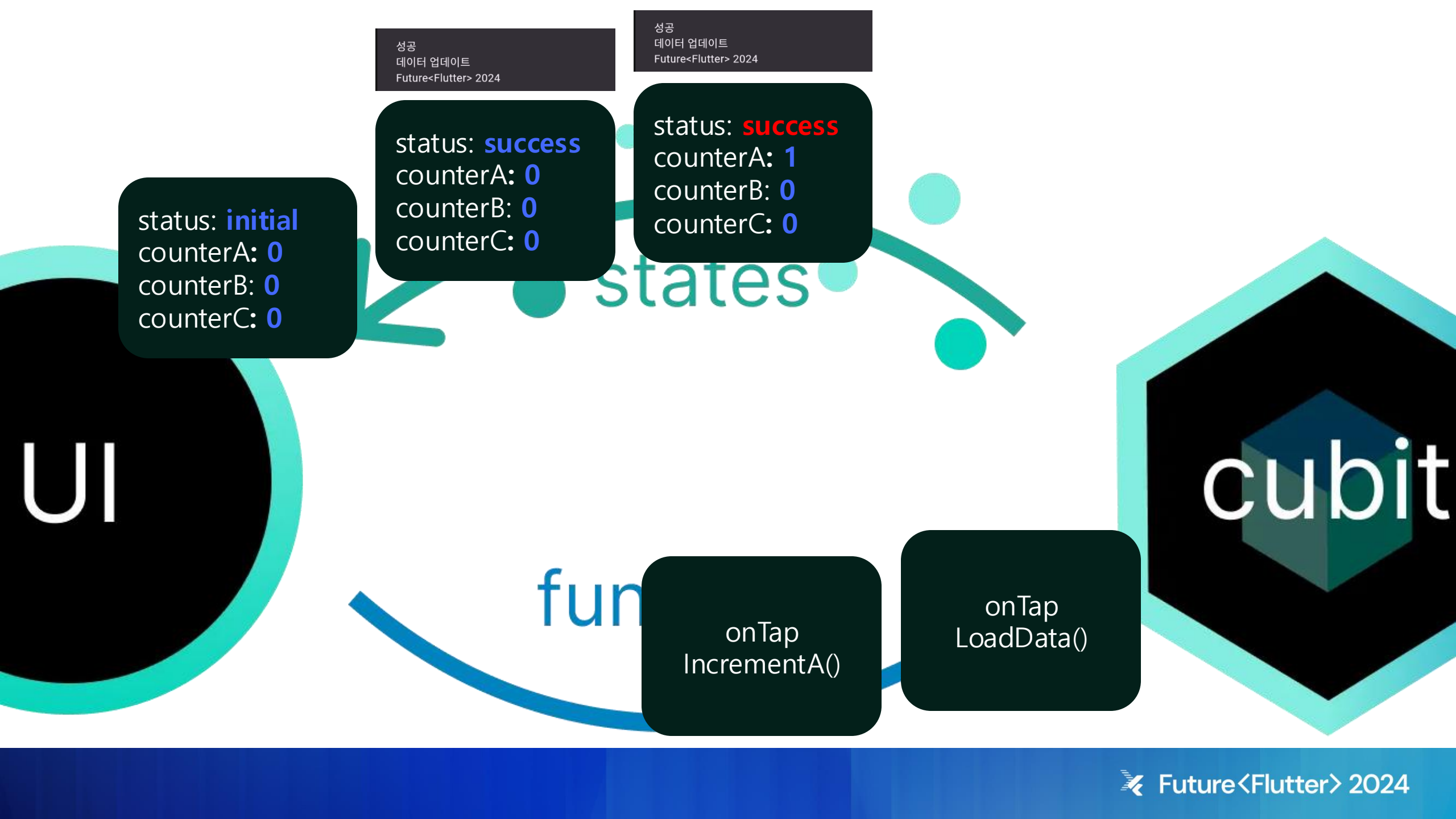
스낵바는 왜 보입니까?

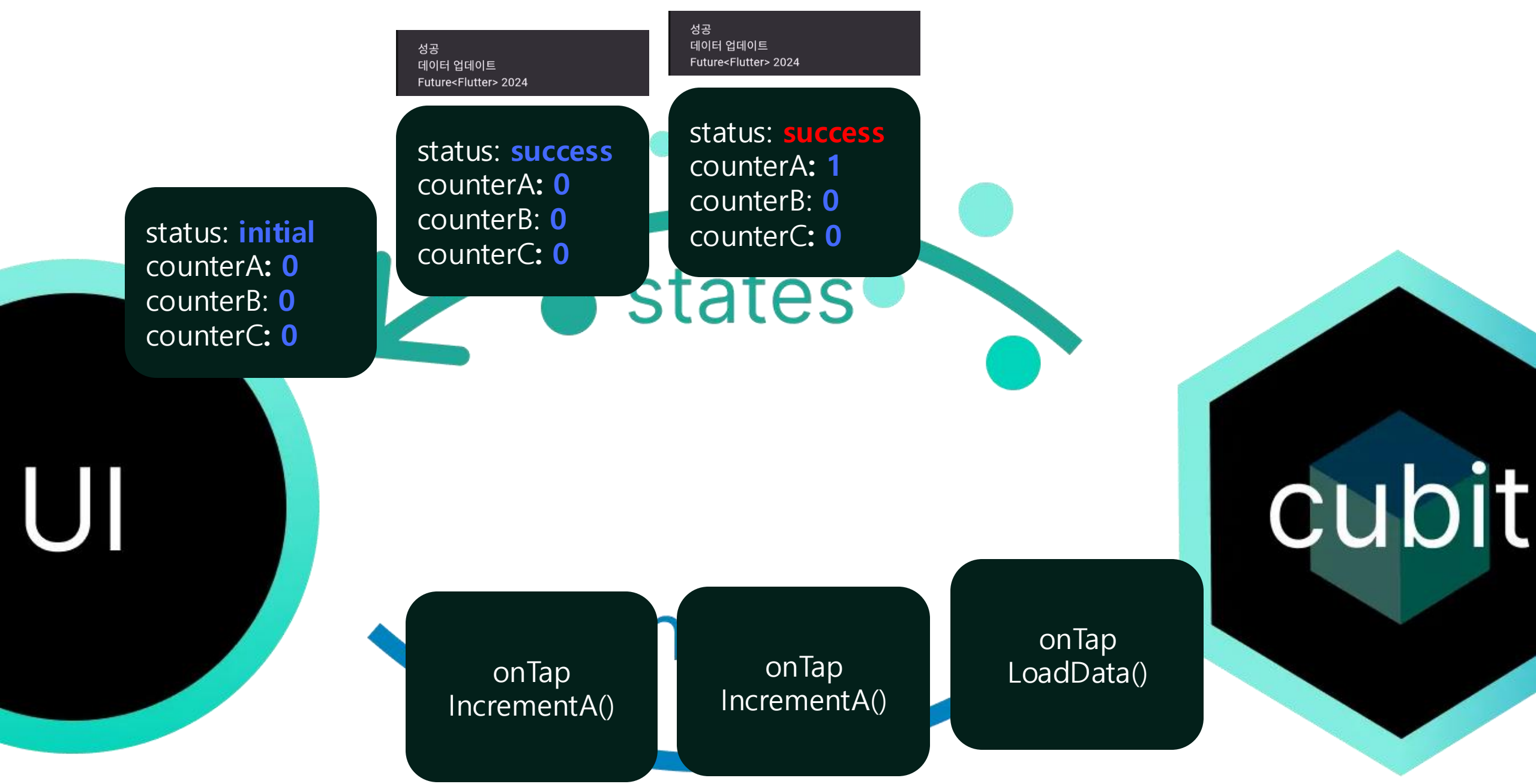

status: initial
counterA: 0
counterB: 0
counterC: 0
states
UI
functions
cubit

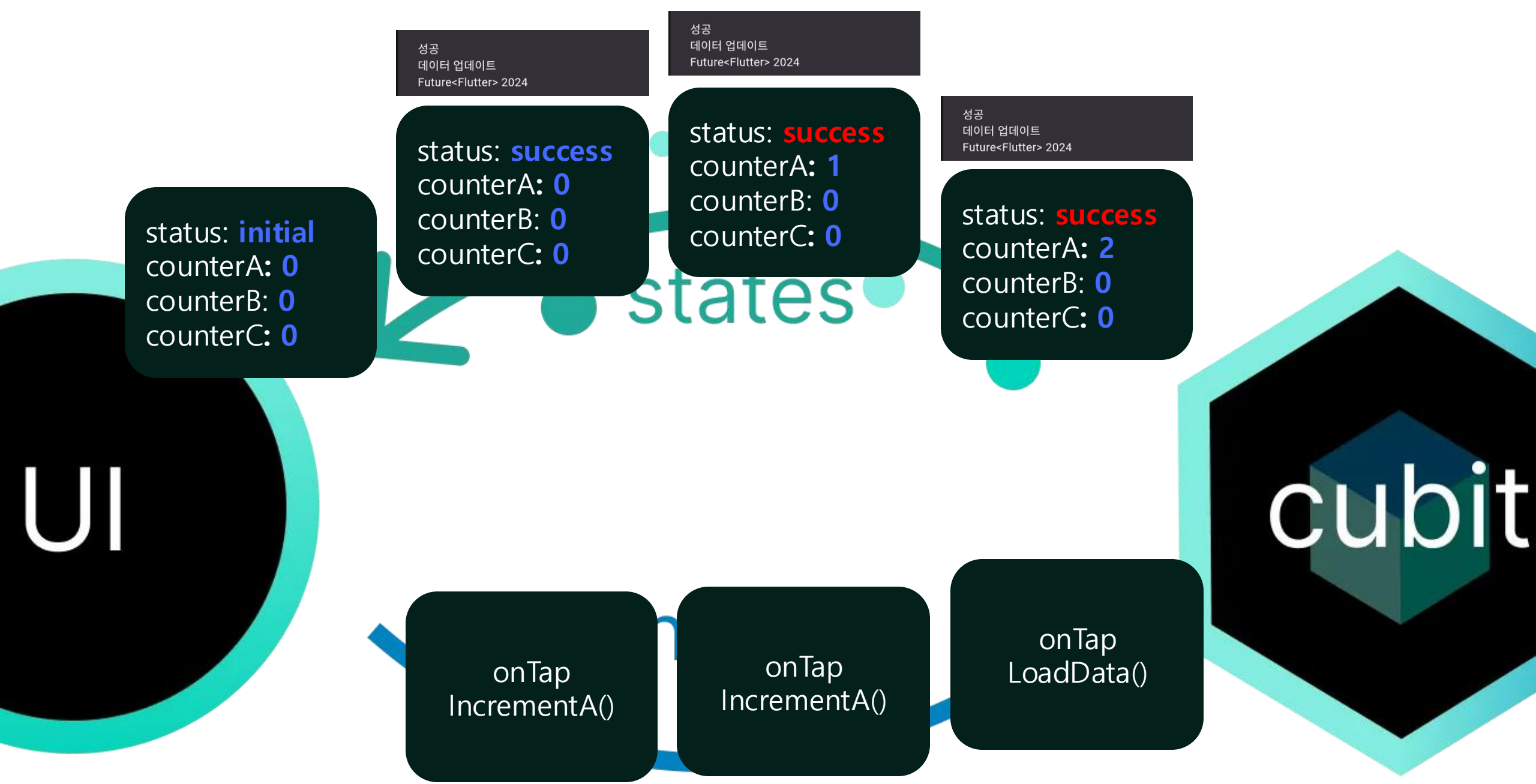

status: initial
counterA: 0
counterB: 0
counterC: 0
states
UI
cubit
functions
onTap
LoadData()


성공
데이터 업데이트
Future<Flutter> 2024
status: success
counterA: 0
counterB: 0
counterC: 0
status: initial
counterA: 0
counterB: 0
counterC: 0
states
UI
cubit
functions
onTap
LoadData()


성공
데이터 업데이트
Future<Flutter> 2024
status: success
counterA: 0
counterB: 0
counterC: 0
status: initial
counterA: 0
counterB: 0
counterC: 0
states
UI
cubit
fun
onTap
IncrementA()
onTap
LoadData()


성공
데이터 업데이트
Future<Flutter> 2024
성공
데이터 업데이트
Future<Flutter> 2024
status: initial
counterA: 0
counterB: 0
counterC: 0
status: success
counterA: 0
counterB: 0
counterC: 0
status: success
counterA: 1
counterB: 0
counterC: 0
states
UI
cubit
fun
onTap
IncrementA()
onTap
LoadData()


성공
데이터 업데이트
Future<Flutter> 2024
성공
데이터 업데이트
Future<Flutter> 2024
status: initial
counterA: 0
counterB: 0
counterC: 0
status: success
counterA: 0
counterB: 0
counterC: 0
status: success
counterA: 1
counterB: 0
counterC: 0
states
UI
cubit
onTap
IncrementA()
onTap
IncrementA()
onTap
LoadData()


성공
데이터 업데이트
Future<Flutter> 2024
성공
데이터 업데이트
Future<Flutter> 2024
성공
데이터 업데이트
Future<Flutter> 2024
status: initial
counterA: 0
counterB: 0
counterC: 0
status: success
counterA: 0
counterB: 0
counterC: 0
status: success
counterA: 1
counterB: 0
counterC: 0
status: success
counterA: 2
counterB: 0
counterC: 0
states
UI
cubit
onTap
IncrementA()
onTap
IncrementA()
onTap
LoadData()

```dart
enum OneTimeNgStatus { initial, loading, success }


void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
  );
  emit(newState);
}
```

```dart
enum OneTimeNgStatus { initial, loading, success }


void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
  );
  emit(newState);
}
```

반복 문제 해결하려면

success가 아닌 status로 변경 필요

```dart
enum OneTimeNgStatus { initial, loading, success }
```

```dart
void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
  );
  emit(newState);
}
```

```dart
void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
    status: OneTimeNgStatus.initial,
  );
  emit(newState);
}
```

```dart
void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
    status: OneTimeNgStatus.loading,
  );
  emit(newState);
}
```

```dart
enum OneTimeNgStatus { initial, loading, success }
```

```dart
void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
  );
  emit(newState);
}
```

지금의 설계로는

해결이 어렵습니다

```dart
void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
    status: OneTimeNgStatus.initial,
  );
  emit(newState);
}
```

```dart
void onTapIncrementA() {
  final newState = state.copyWith(
    counterA: state.counterA + 1,
    status: OneTimeNgStatus.loading,
  );
  emit(newState);
}
```

다이얼로그

다음 화면

이전 화면

다이얼로그

다음 화면

이전 화면

```
enum OneTimeNg2Status {
  initial,
  loading,
  success,

}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,


  );
```

```dart
void _showDialog(String message) {
  emit(
    state.copyWith(
      status: OneTimeNg2Status.showDialog,
      dialogMessage: message,
    ),
  );
}
```

다음 화면

이전 화면

```dart
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,

}
```

```dart
class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,

  );
```

```dart
void _showDialog(String message) {
  emit(
    state.copyWith(
      status: OneTimeNg2Status.showDialog,
      dialogMessage: message,
    ),
  );
}

void _goToNextPage() {
  emit(
    state.copyWith(
      status: OneTimeNg2Status.goToNextPage,
      nextPageData: 200,
    ),
  );
}
```

이전 화면

```dart
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,
  goToNextPage,
}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,
    this.nextPageData,
  );
```

```dart
void _showDialog(String message) {
  emit(
    state.copyWith(
      status: OneTimeNg2Status.showDialog,
      dialogMessage: message,
    ),
  );
}

void _goToNextPage() {
  emit(
    state.copyWith(
      status: OneTimeNg2Status.goToNextPage,
      nextPageData: 200,
    ),
  );
}

void _goToPrevPage() {
  emit(state.copyWith(
    status: OneTimeNg2Status.goToPrevPage,
    prevPageData: -1,
  ));
}
```

```dart
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,
  goToNextPage,
  goToPrevPage,
}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,
    this.nextPageData,
    this.prevPageData,
  );
```

논리적 충돌

```
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,
  goToNextPage,
  goToPrevPage,
}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,
    this.nextPageData,
    this.prevPageData,
  );
```

```dart
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,
  goToNextPage,
  goToPrevPage,
}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,
    this.nextPageData,
    this.prevPageData,
  );
```

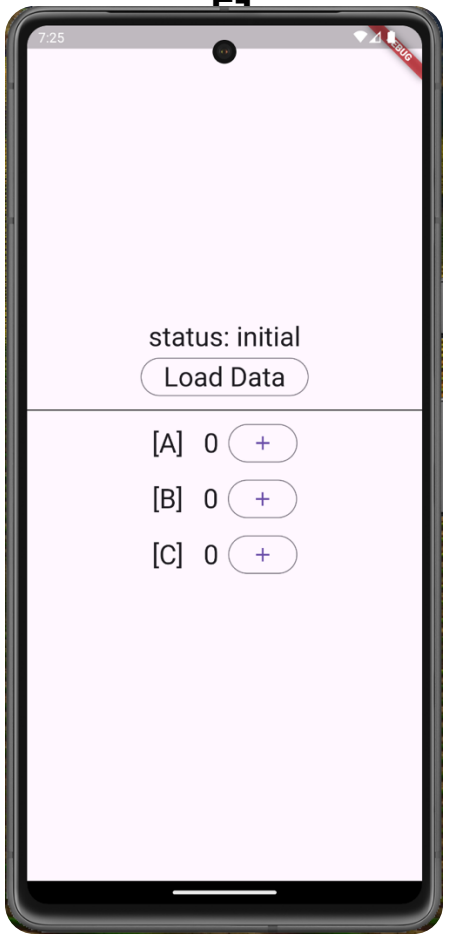
불필요한 data

emit 호출 시 반복적 실행

status enum 논리적 충돌

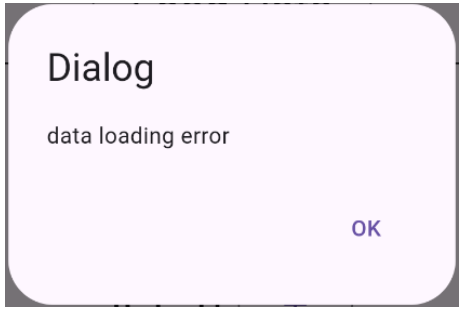
state class 불필요한 data

화면을 그리는 데이터




```dart
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,
  goToPrevPage,
  goToNextPage,
}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,
    this.prevPageData,
    this.nextPageData,
  );
```

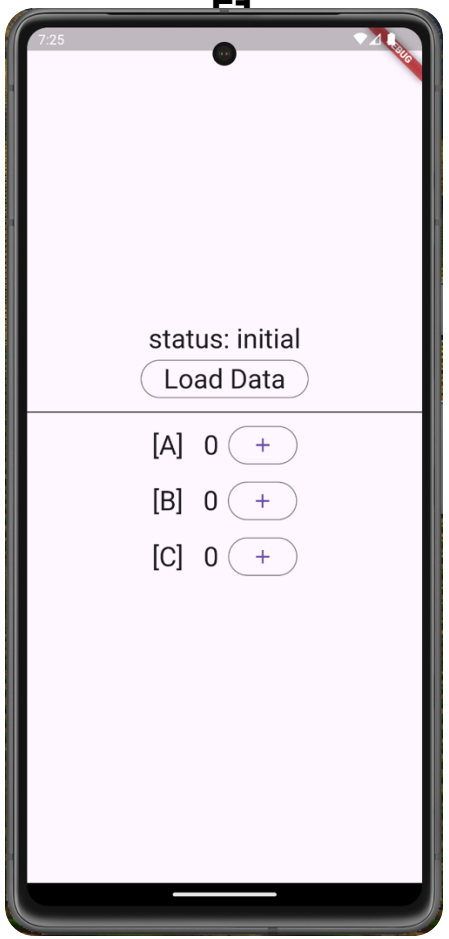
일회성 동작 데이터



다음 화면

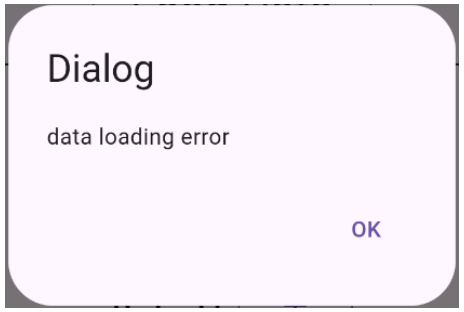
이전 화면

화면을 그리는 데이터




```
enum OneTimeNg2Status {
  initial,
  loading,
  success,
  showDialog,
  goToPrevPage,
  goToNextPage,
}

class OneTimeNg2State {
  const OneTimeNg2State(
    this.status,
    this.counterA,
    this.counterB,
    this.counterC,
    this.dialogMessage,
    this.prevPageData,
    this.nextPageData,
  );
```

일회성 동작 데이터



다음 화면

이전 화면

```dart
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class Empty

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class Empty

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class ShowDialog
String dialogMessage

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class Empty

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class ShowDialog
String dialogMessage

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class GoToNextPage
int nextPageData

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class ShowSnackBar

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

class GoToPrevPage
int prevPageData

```
class OneTimeNgState {
  final OneTimeNgStatus status;
  final int counterA;
  final int counterB;
  final int counterC;
```

화면을 그리는 데이터

일회성 동작 데이터

중첩된 클래스로 구분

살짝 복잡해 보이는데요

코드는 어떻게 합니까

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

논리적 충돌 해결

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

논리적 충돌 해결

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

불필요한 data 해결

```dart
// 일회성 동작 데이터
@freezed
sealed class OneTimeState with _$OneTimeState {
  const factory OneTimeState.empty(OneTimeData data) = Empty;
  const factory OneTimeState.showSnackBar(OneTimeData data) = ShowSnackBar;
  const factory OneTimeState.showDialog(OneTimeData data, String dialogMessage) = ShowDialog;
  const factory OneTimeState.goToPrevPage(OneTimeData data, int prevPageData) = GoToPrevPage;
  const factory OneTimeState.goToNextPage(OneTimeData data, int nextPageData) = GoToNextPage;
}

// 화면을 그리는 데이터
@freezed
class OneTimeData with _$OneTimeData {
  const factory OneTimeData({
    @Default(OneTimeStatus.initial) OneTimeStatus status,
    @Default(0) int counterA,
    @Default(0) int counterB,
    @Default(0) int counterC,
  }) = _OneTimeData;
}

enum OneTimeStatus { initial, loading, success }
```

불필요한 data 해결

```dart
void onTapIncrementA() {
  emit(
    OneTimeState.empty(
      state.data.copyWith(counterA: state.data.counterA + 1),
    ),
  );
}
```

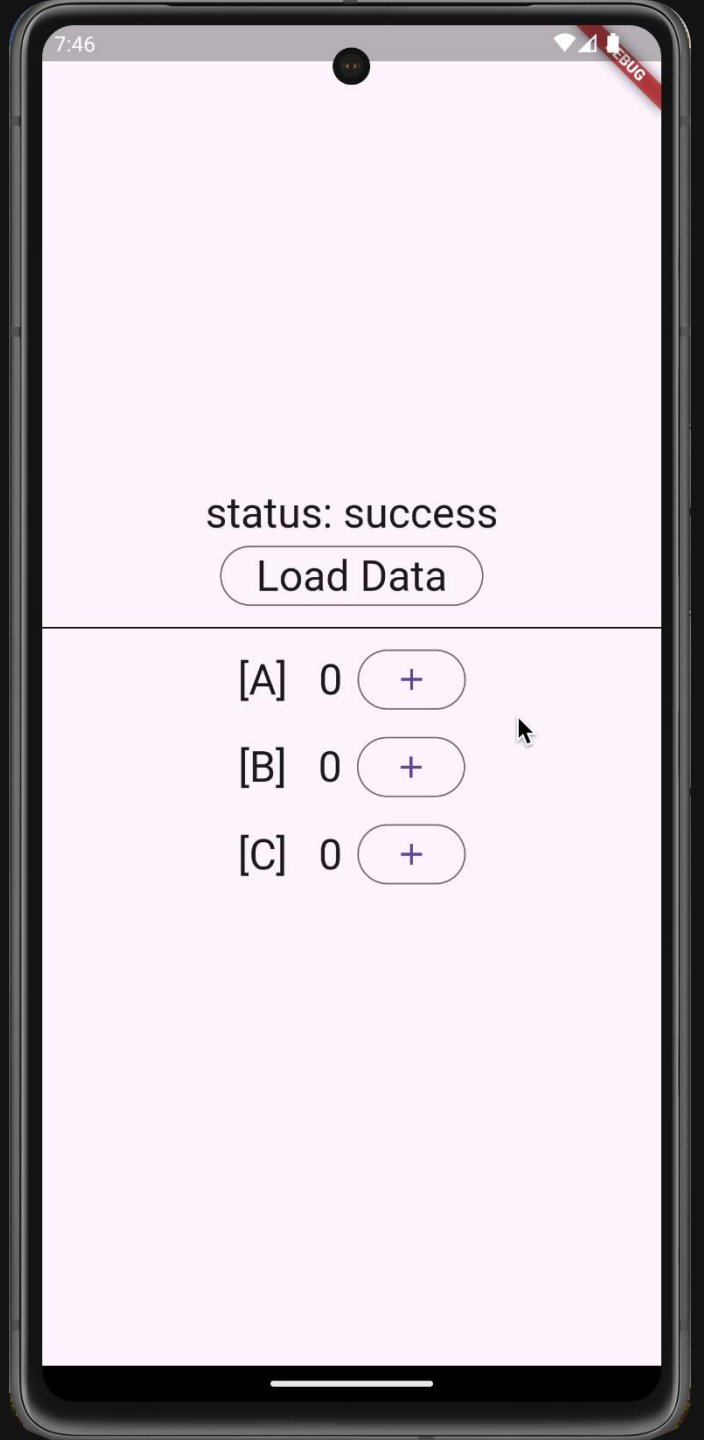
```dart
void onTapIncrementA() {
  emit(
    OneTimeState.empty(
      state.data.copyWith(counterA: state.data.counterA + 1),
    ),
  );
}
```

반복적 실행 해결

~~반복적 실행~~ 해결

~~논리적 충돌~~ 해결

~~불필요한 data~~ 해결


7:46
DEBUG
status: success
Load Data
[A] 0 +
[B] 0 +
[C] 0 +

그런데...

코드가 길어져서

코드 작성이

부담스럽습니다

3. 코드 작성 피로도 줄이기

작성할 최소한의 코드

파일 3개

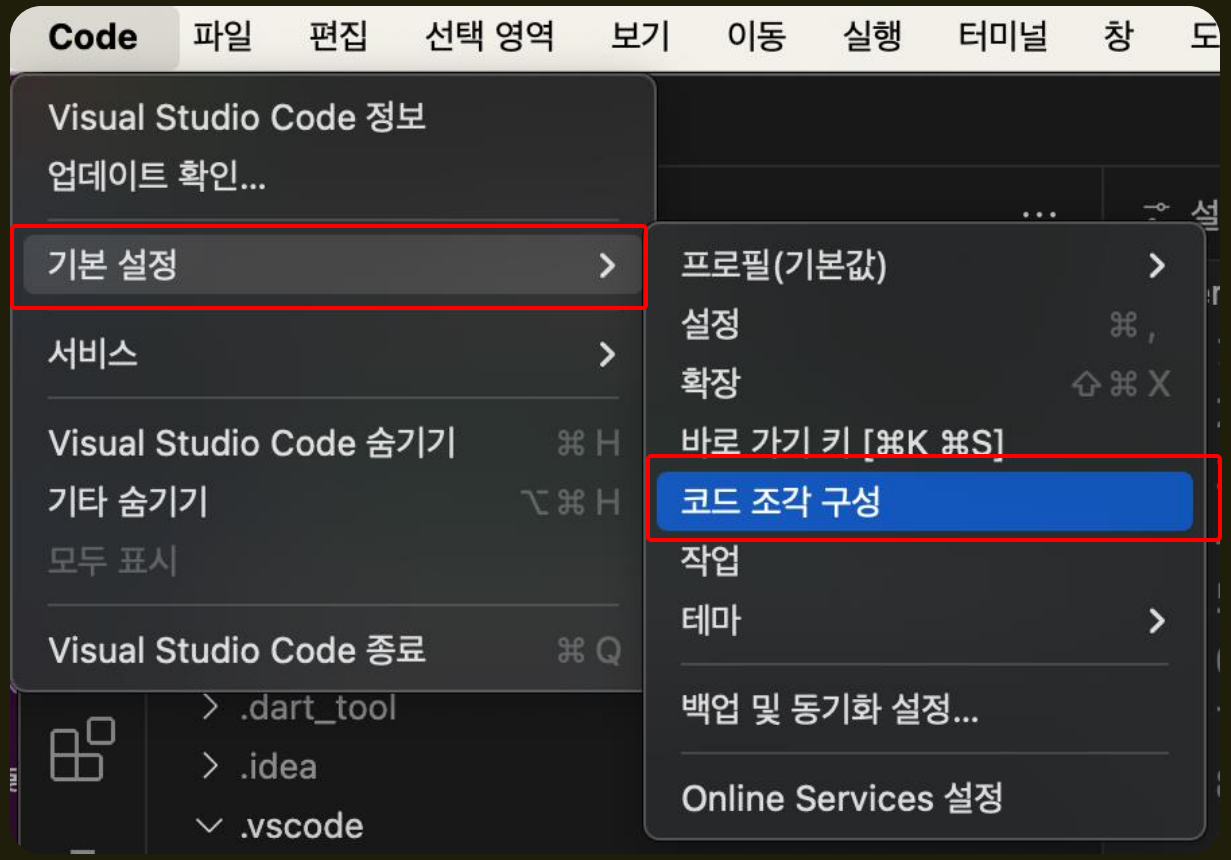
60줄

1200자

VS Code: Snippets

IntelliJ IDEA: Live Templates

Visual Studio Code



```
  "myState": {
    "scope": "dart",
    "prefix": "myState",
    "body": [
      "import 'package:freezed_annotation/freezed_annotation.dart';",
      "",
      "part '$2_state.freezed.dart';",
      "",
      "@freezed",
      "sealed class $1State with _$$1State {",
      "  const factory $1State.empty($1Data data) = Empty;",
      "}",
      "",
      "@freezed",
      "class $1Data with _$$1Data {",
      "  const factory $1Data({",
      "    @Default('') String sample,",
      "  }) = _$1Data;",
      "}",
    ],
  },
```

```
  "myState": {
    "scope": "dart",
    "prefix": "myState",
    "body": [
      "import 'package:freezed_annotation/freezed_annotation.dart';",
      "",
      "part '$2_state.freezed.dart';",
      "",
      "@freezed",
      "sealed class $1State with _$$1State {",
      "  const factory $1State.empty($1Data data) = Empty;",
      "}",
      "",
      "@freezed",
      "class $1Data with _$$1Data {",
      "  const factory $1Data({",
      "    @Default('') String sample,",
      "  }) = _$1Data;",
      "}",
    ],
  },
```

```
  "myState": {
    "scope": "dart",
    "prefix": "myState",
    "body": [
      "import 'package:freezed_annotation/freezed_annotation.dart';",
      "",
      "part '$2_state.freezed.dart';",
      "",
      "@freezed",
      "sealed class $1State with _$$1State {",
      "  const factory $1State.empty($1Data data) = Empty;",
      "}",
      "",
      "@freezed",
      "class $1Data with _$$1Data {",
      "  const factory $1Data({",
      "    @Default('') String sample,",
      "  }) = _$1Data;",
      "}",
    ],
  },
```

```
3   "myState": {
4     "scope": "dart",
5     "prefix": "myState",
6     "body": [
7       "import 'package:freezed_annotation/freezed_annotation.dart';",
8       "",
9       "part '$2_state.freezed.dart';",
10      "",
11      "@freezed",
12      "sealed class $1State with _$$1State {",
13      "  const factory $1State.empty($1Data data) = Empty;",
14      "}",
15      "",
16      "@freezed",
17      "class $1Data with _$$1Data {",
18      "  const factory $1Data({",
19      "    @Default('') String sample,",
20      "  }) = _$1Data;",
21      "}",
22    ],
23  },
```

```
"mycubit": {
    "scope": "dart",
    "prefix": "mycubit",
    "body": [
        "import 'package:flutter_bloc/flutter_bloc.dart';",
        "",
        "class $1Cubit extends Cubit<$1State> {",
        "  $1Cubit() : super(_initState);",
        "",
        "  static $1State get _initState => const $1State.empty($1Data());",
        "",
        "  void onTapXxx() {",
        "    emit($1State.empty(state.data));",
        "  }",
        "}",
    ],
},
```

```
"myview": {
    "scope": "dart",
    "prefix": "myview",
    "body": [
        "import 'package:flutter/material.dart';",
        "import 'package:flutter_bloc/flutter_bloc.dart';",
        "",
        "class $1BlocView extends StatelessWidget {",
        "  const $1BlocView({super.key});",
        "",
        "  @override",
        "  Widget build(BuildContext context) {",
        "    return BlocProvider(",
        "      create: (_) => $1Cubit(),",
        "      child: BlocListener<$1Cubit, $1State>(",
        "        listener: (context, state) {",
        "          switch (state) {",
        "            case Empty():",
        "              break;",
        "          }",
        "        },",
        "        child: const $1View(),",
        "      ),",
        "    );",
        "  }",
        "}",
        "",
        "class $1View extends StatelessWidget {",
        "  const $1View({super.key});",
        "",
        "  @override",
        "  Widget build(BuildContext context) {",
        "    return const Placeholder($2);",
        "  }",
        "}",
```

Snippets 작성 완료 !!!

잘 되는지 확인해 봅시다

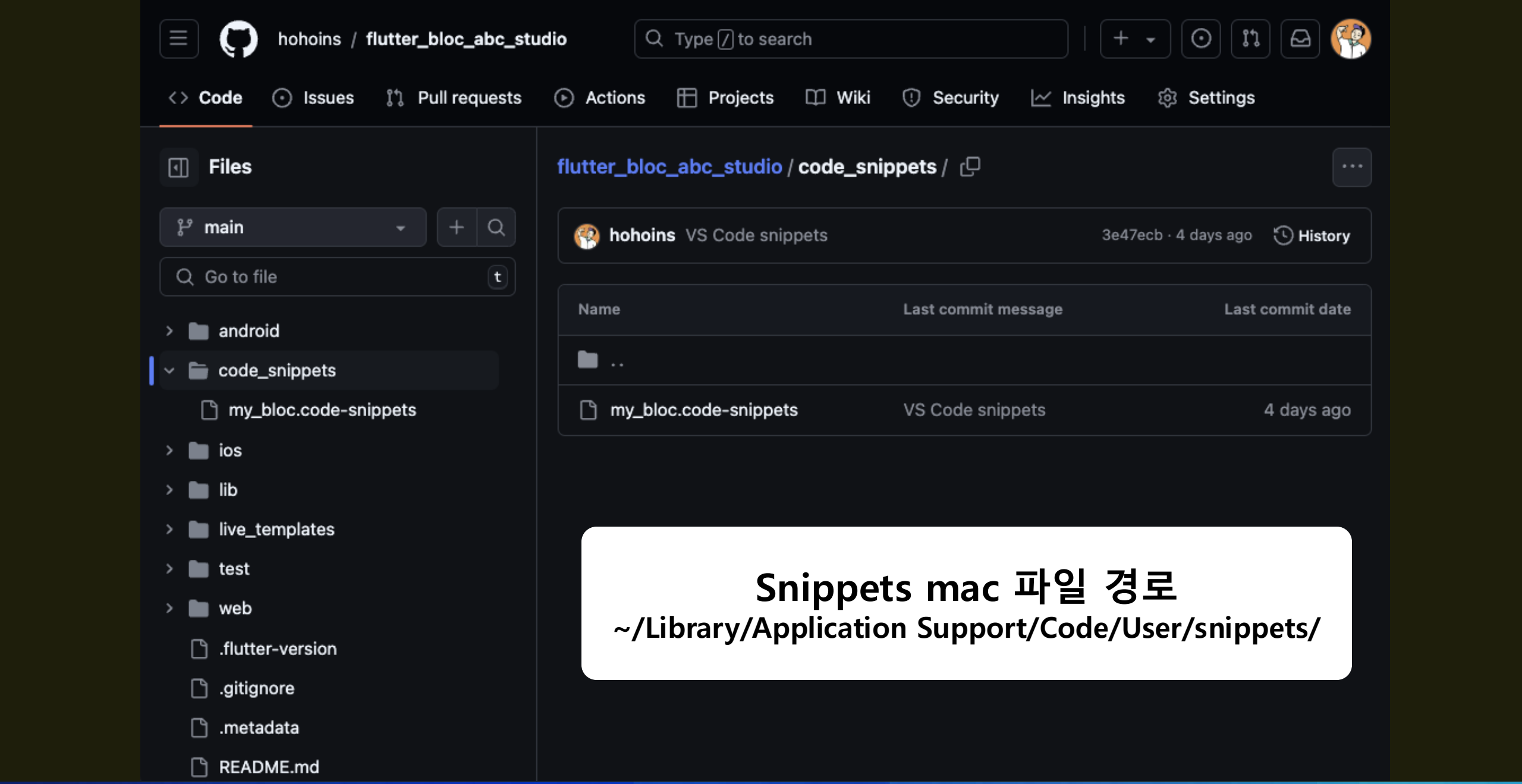
my_bloc.code-snippets | splash_state.dart M | splash_cubit.dart M | splash_view.dart M

Users > user > Library > Application Support > Code > User > snippets > my_bloc.code-snippets > {} myState > [ ] body

```
{

    "myState": {
        "scope": "dart",
        "prefix": "myState",
        "body": [
            "import 'package:freezed_annotation/freezed_annotation.dart';",
            "",
            "part '$2_state.freezed.dart';",
            "",
            "@freezed",
            "sealed class $1State with _$$1State {",
            "  const factory $1State.empty($1Data data) = Empty;",
            "}",
            "",
            "@freezed",
            "class $1Data with _$$1Data {",
            "  const factory $1Data({",
            "    @Default('') String sample,",
            "  }) = _$1Data;",
            "}",
        ],
    },

    "mycubit": {
        "scope": "dart",
        "prefix": "mycubit",
        "body": [
            "import 'package:flutter_bloc/flutter_bloc.dart';",
            "",
            "class $1Cubit extends Cubit<$1State> {",
            "  $1Cubit() : super(_initState);",
            "",
            "  static $1State get _initState => const $1State.empty($1Data());",
            "",
            "  void onTapXxx() {",
            "    emit($1State.empty(state.data));",
```

이렇게 좋은 것은

반드시 팀원들과 공유!!!



Snippets mac 파일 경로

~/Library/Application Support/Code/User/snippets/

https://github.com/hohoins/flutter_bloc_abc_studio/tree/main/code_snippets

템플릿 처리 1석 3조

코딩 피로도 감소

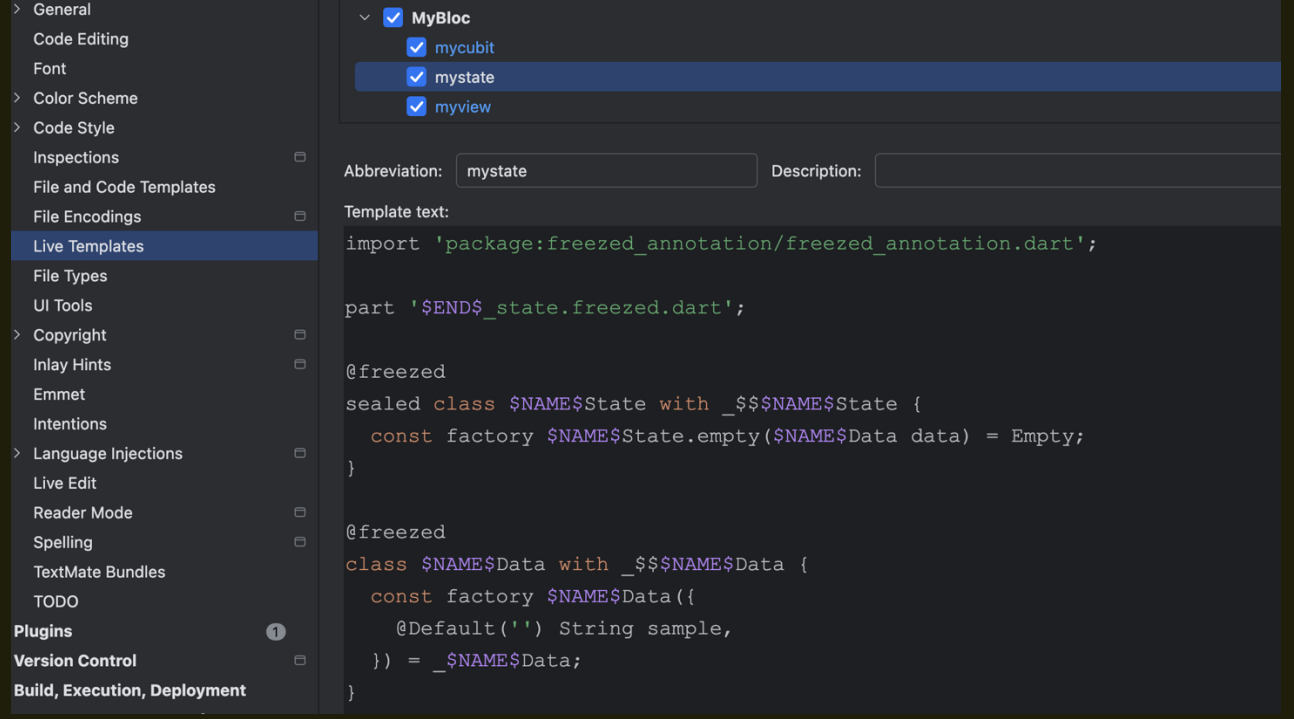
일관된 코딩 컨벤션

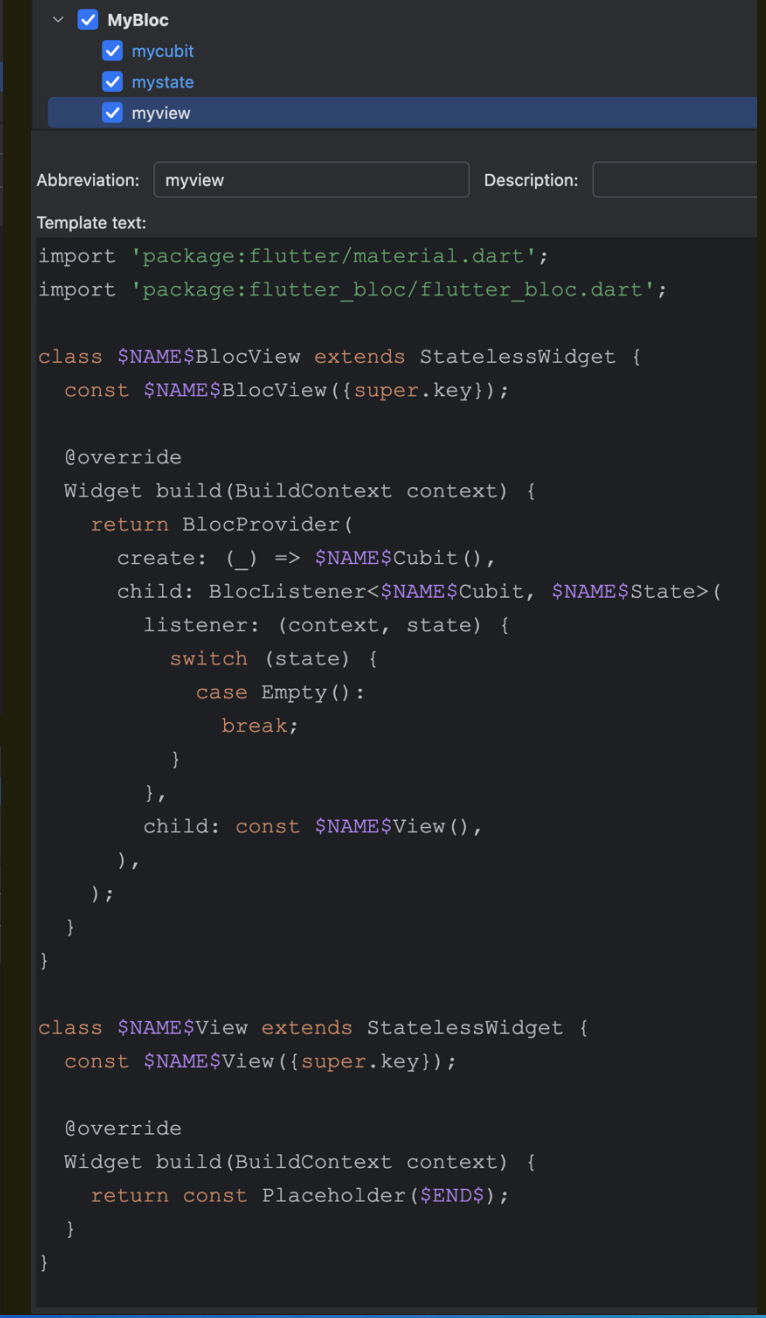
작업 속도 향상

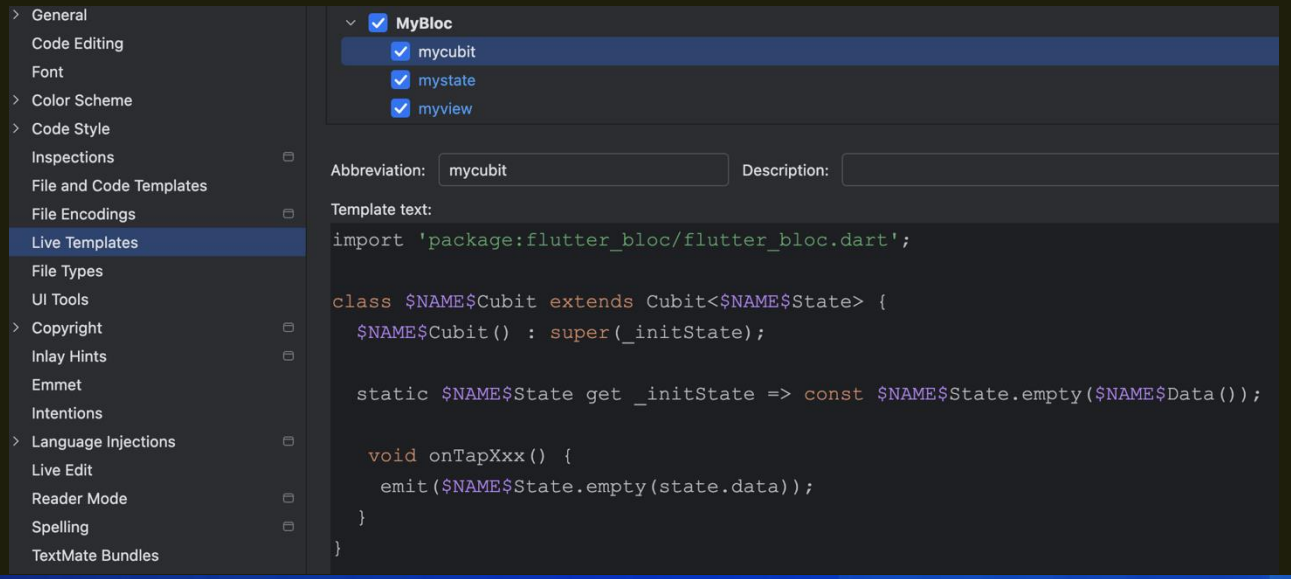
IntelliJ IDEA

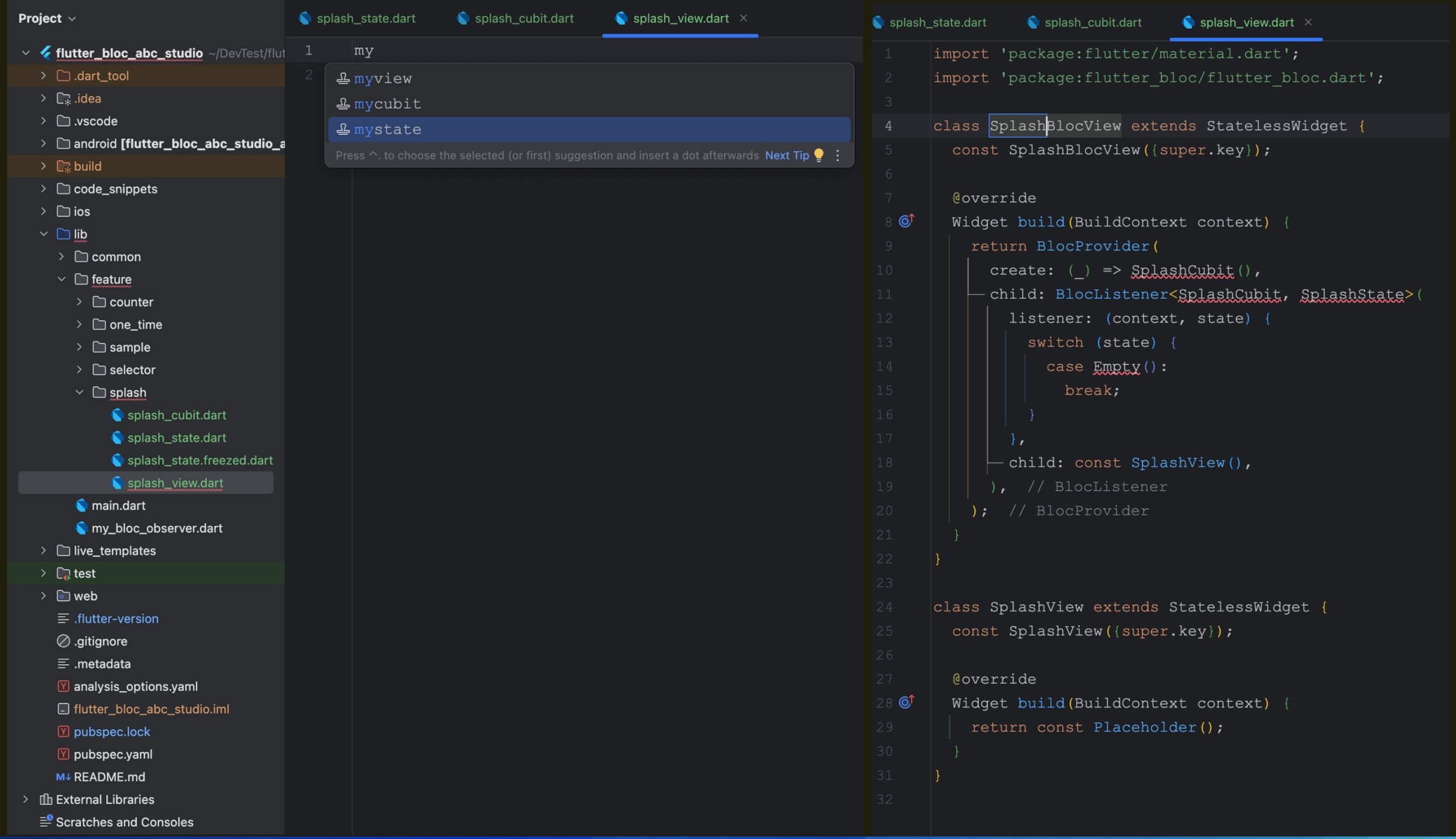
Android Studio

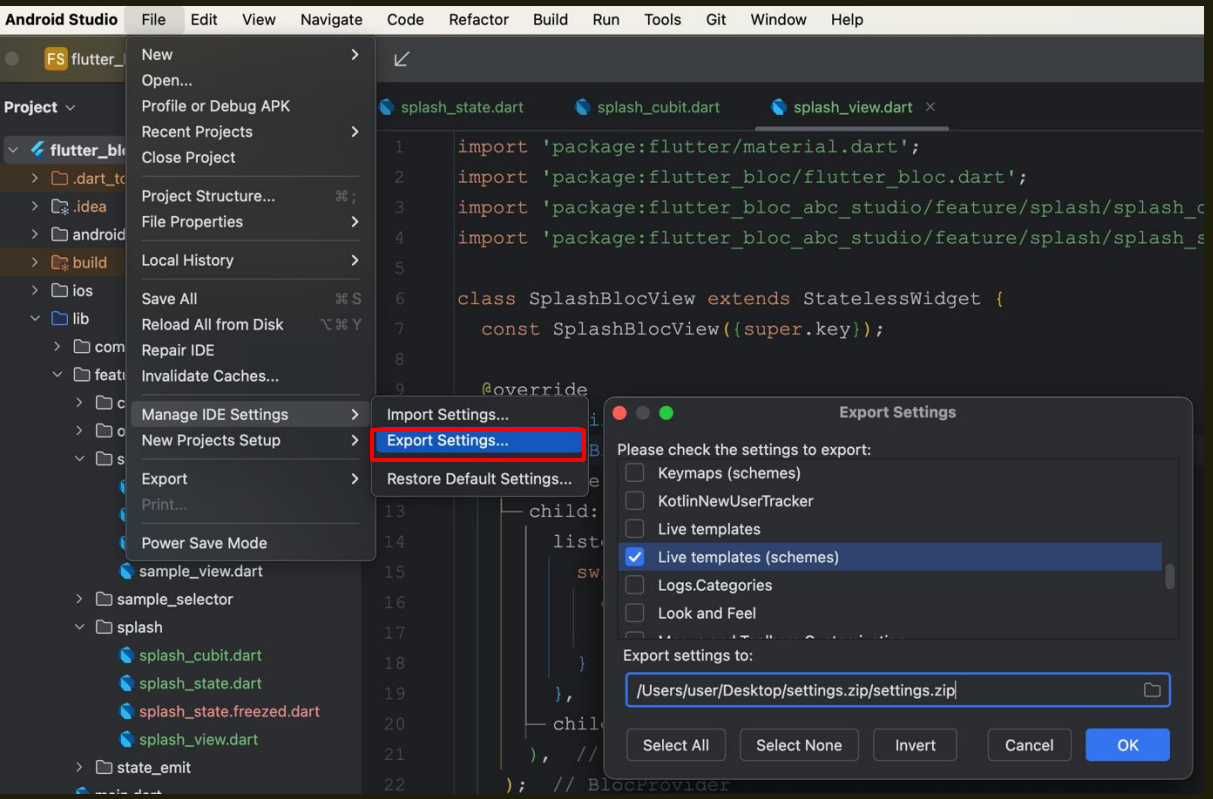
Settings> Editor> Live Templates

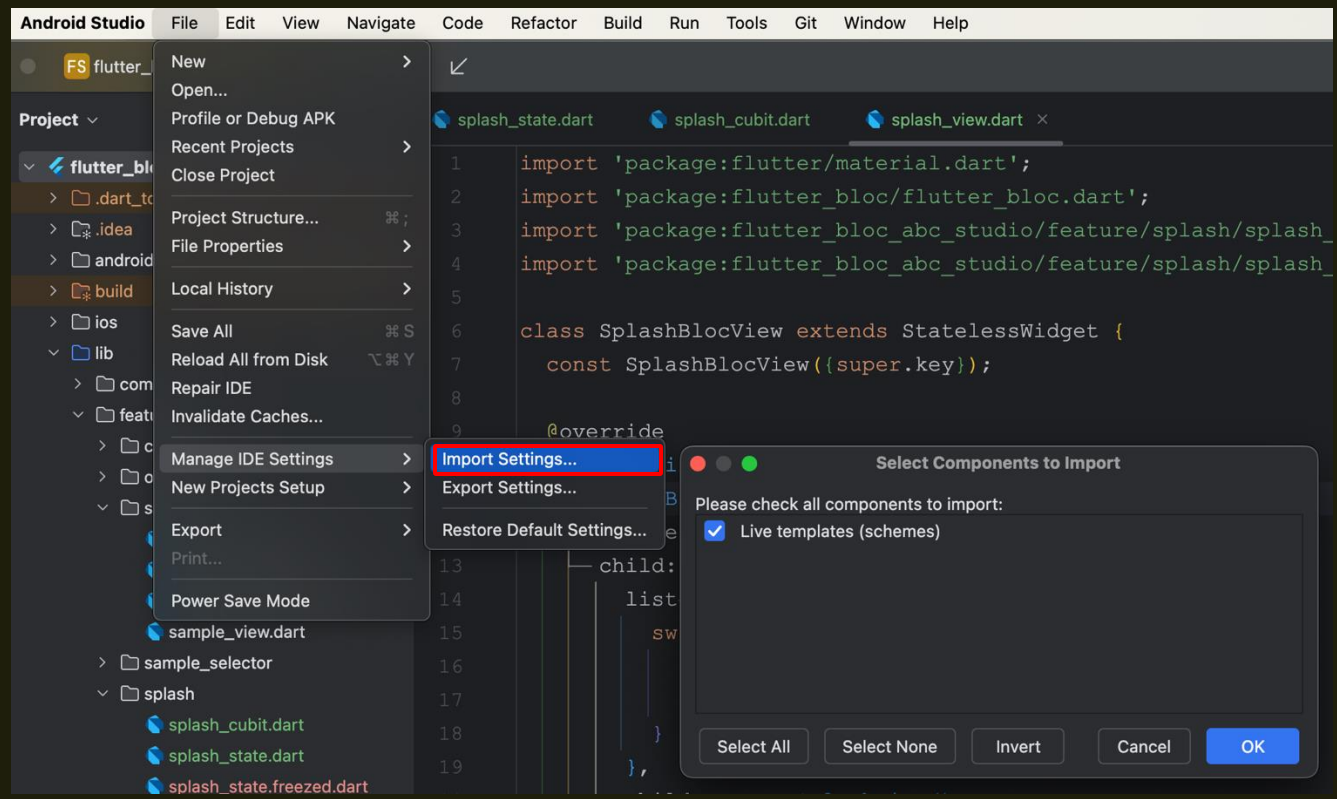

General
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
UI Tools
Copyright
Inlay Hints
Emmet
Intentions
Language Injections
Live Edit
Reader Mode
Spelling
TextMate Bundles
TODO
Plugins
Version Control
Build, Execution, Deployment
MyBloc
mycubit
mystate
myview
Abbreviation: mystate
Description:
Template text:
import 'package:freezed_annotation/freezed_annotation.dart';
part 'END_state.freezed.dart';
@freezed
sealed class $NAME$State with _$$$NAME$State {
const factory $NAME$State.empty($NAME$Data data) = Empty;
}
@freezed
class $NAME$Data with _$$$NAME$Data {
const factory $NAME$Data({
@Default('') String sample,
}) = _$NAME$Data;
}


MyBloc
mycubit
mystate
myview
Abbreviation: myview
Description:
Template text:
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
class $NAME$BlocView extends StatelessWidget {
const $NAME$BlocView({super.key});
@override
Widget build(BuildContext context) {
return BlocProvider(
create: (_) => $NAME$Cubit(),
child: BlocListener<$NAME$Cubit, $NAME$State>(
listener: (context, state) {
switch (state) {
case Empty():
break;
}
},
child: const $NAME$View(),
),
);
}
}
class $NAME$View extends StatelessWidget {
const $NAME$View({super.key});
@override
Widget build(BuildContext context) {
return const Placeholder(END);
}
}


General
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
UI Tools
Copyright
Inlay Hints
Emmet
Intentions
Language Injections
Live Edit
Reader Mode
Spelling
TextMate Bundles
MyBloc
mycubit
mystate
myview
Abbreviation: mycubit
Description:
Template text:
import 'package:flutter_bloc/flutter_bloc.dart';
class $NAME$Cubit extends Cubit<$NAME$State> {
$NAME$Cubit() : super(_initState);
static $NAME$State get _initState => const $NAME$State.empty($NAME$Data());
void onTapXxx() {
emit($NAME$State.empty(state.data));
}
}


Project
flutter_bloc_abc_studio ~/DevTest/flut
.dart_tool
.idea
.vscode
android [flutter_bloc_abc_studio_a
build
code_snippets
ios
lib
common
feature
counter
one_time
sample
selector
splash
splash_cubit.dart
splash_state.dart
splash_state.freezed.dart
splash_view.dart
main.dart
my_bloc_observer.dart
live_templates
test
web
.flutter-version
.gitignore
.metadata
analysis_options.yaml
flutter_bloc_abc_studio.iml
pubspec.lock
pubspec.yaml
README.md
External Libraries
Scratches and Consoles
splash_state.dart
splash_cubit.dart
splash_view.dart
my
myview
mycubit
mystate
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards
Next Tip
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
class SplashBlocView extends StatelessWidget {
const SplashBlocView({super.key});
@override
Widget build(BuildContext context) {
return BlocProvider(
create: (_) => SplashCubit(),
child: BlocListener<SplashCubit, SplashState>(
listener: (context, state) {
switch (state) {
case Empty():
break;
},
child: const SplashView(),
), // BlocListener
); // BlocProvider
}
}
class SplashView extends StatelessWidget {
const SplashView({super.key});
@override
Widget build(BuildContext context) {
return const Placeholder();
}
}

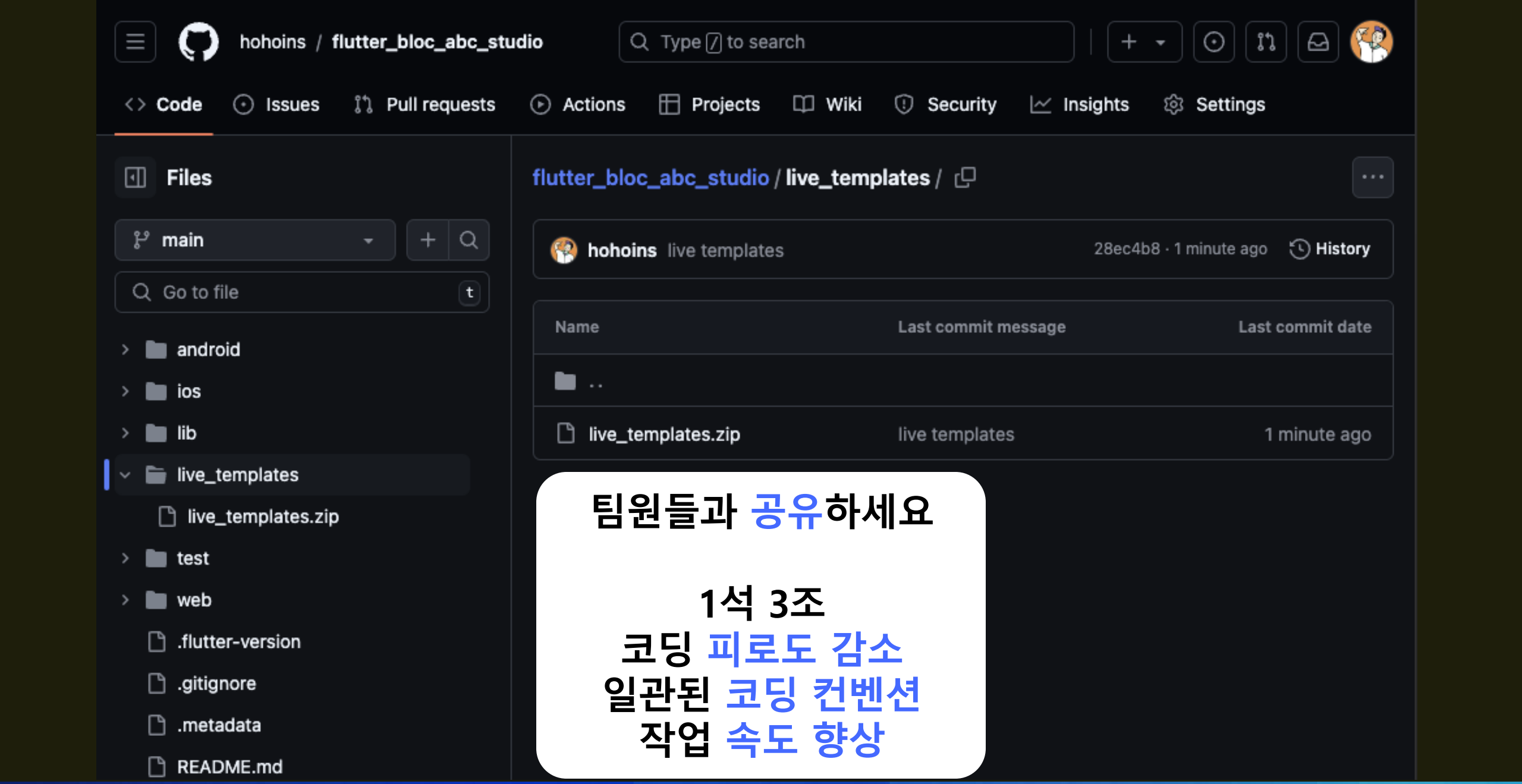




File> Manage> IDE Settings> Export Settings

File> Manage> IDE Settings> Import Setting

Live templates(schemes) 선택



팀원들과 공유하세요

1석 3조
코딩 피로도 감소
일관된 코딩 컨벤션
작업 속도 향상

https://github.com/hohoins/flutter_bloc_abc_studio/tree/main/live_templates

state는 정리되었습니다

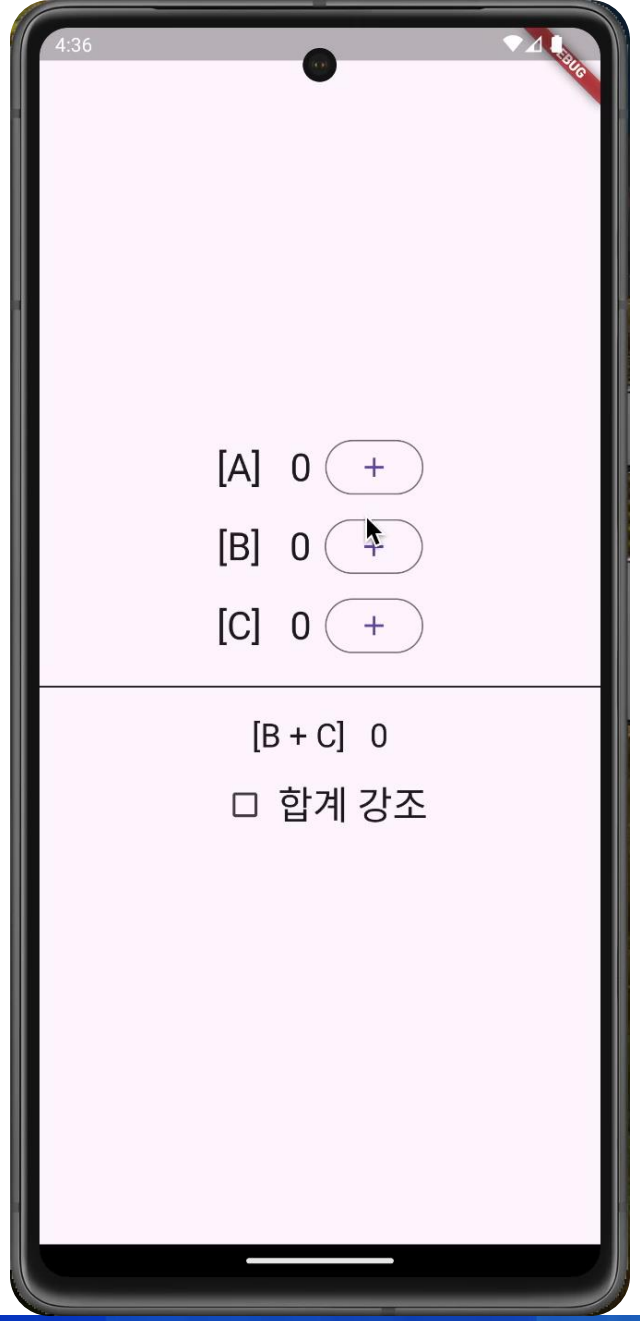
UI 최적화를 해봅시다

4. widget build 최적화

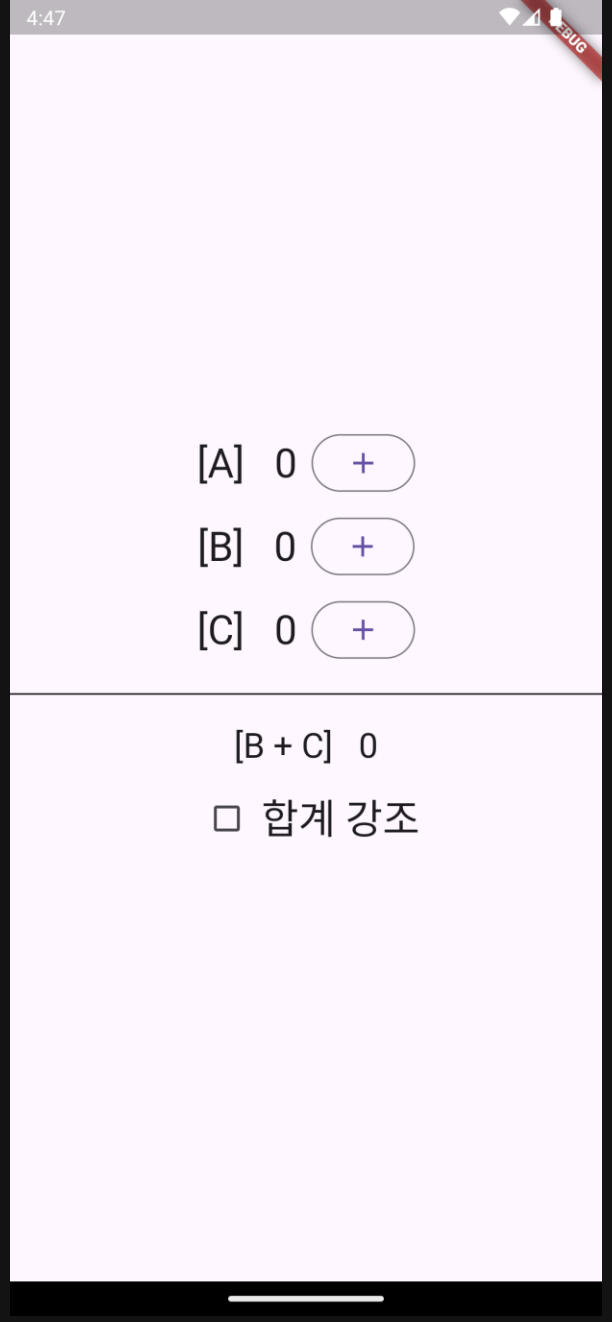
State가 변경되면

대부분의 widget

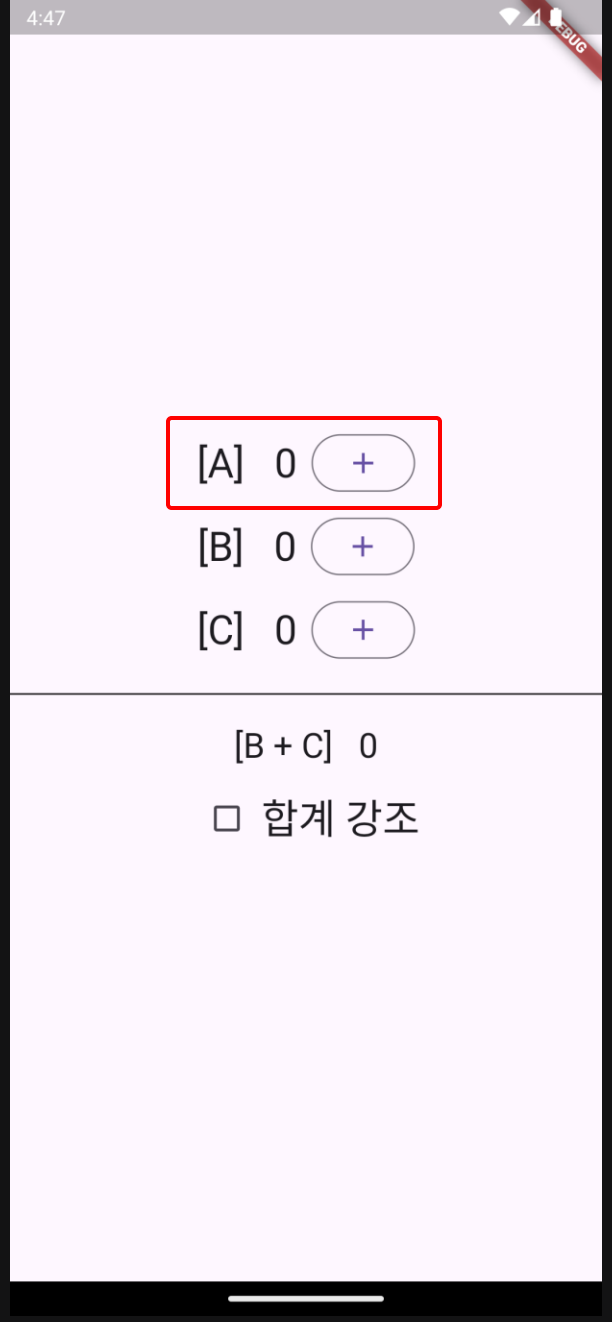
다시 build 되는 문제


4:36
DEBUG
[A] 0 +
[B] 0 +
[C] 0 +
[B + C] 0
합계 강조

원인을 찾아봅시다




```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return CounterWidget(
      text: '[A]    ${state.counterA}',
      onPressed: context.read<SelectorNgCubit>().onTapIncrementA,
    );  // CounterWidget
  },
),  // BlocBuilder
```




```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return CounterWidget(
      text: '[A]    ${state.counterA}',
      onPressed: context.read<SelectorNgCubit>().onTapIncrementA,
    );  // CounterWidget
  },
),  // BlocBuilder
```

참조하는 값이 변경될 때만

build 하고 싶습니다

BlocSelector

BlocSelector는 `BlocBuilder`와 유사하지만 개발자가 현재 bloc state에 따라 새 값을 선택하여 업데이트를 필터링할 수 있는 Flutter 위젯입니다. 선택한 값이 변경되지 않으면 불필요한 빌드가 방지됩니다. `BlocSelector`가 `builder`를 다시 호출해야 하는지 여부를 정확하게 결정하려면 선택한 값을 변경할 수 없어야(immutable) 합니다.

만약 `bloc` 파라미터가 생략되면 `BlocBuilder`는 `BlocProvider`와 현재 `BuildContext`를 사용하여 자동으로 bloc을 조회합니다.

```
BlocSelector<BlocA, BlocAState, SelectedState>(
  selector: (state) {
    // return selected state based on the provided state.
  },
  builder: (context, state) {
    // return widget here based on the selected state.
  },
);
```

https://bloclibrary.dev/flutter-bloc-concepts/#blocselector

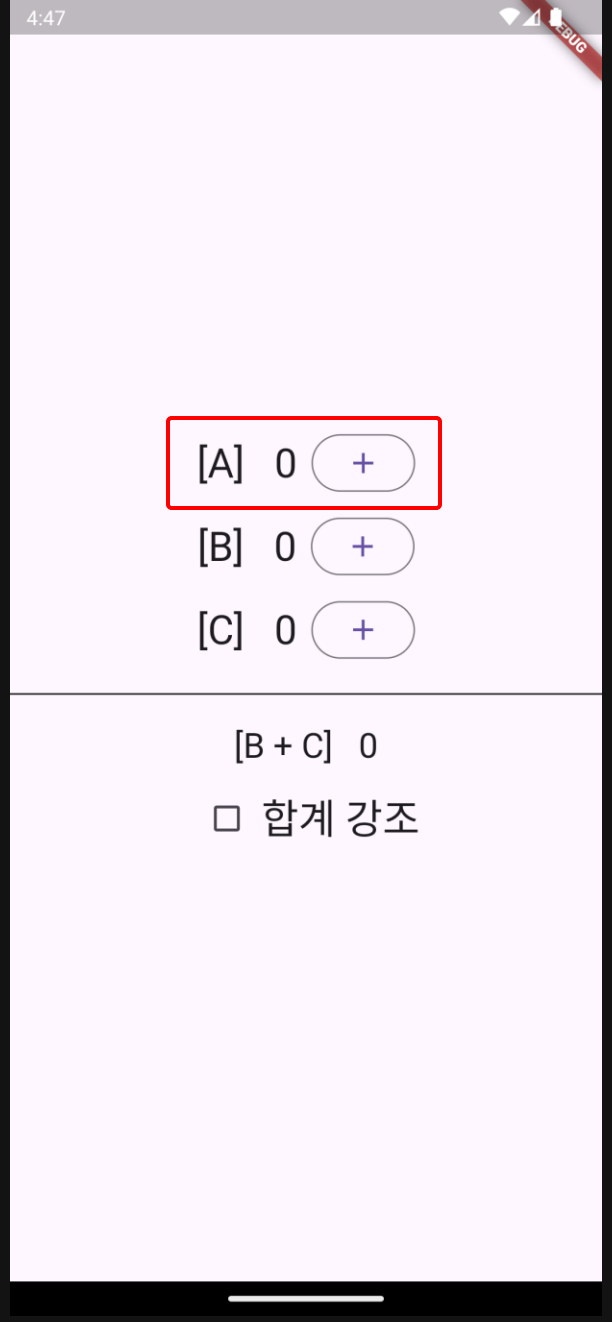
BlocSelector

BlocSelector는 `BlocBuilder`와 유사하지만 개발자가 현재 bloc state에 따라 새 값을 선택하여 업데이트를 필터링할 수 있는 Flutter 위젯입니다. 선택한 값이 변경되지 않으면 불필요한 빌드가 방지됩니다. `BlocSelector`가 `builder`를 다시 호출해야 하는지 여부를 정확하게 결정하려면 선택한 값을 변경할 수 없어야(immutable) 합니다.

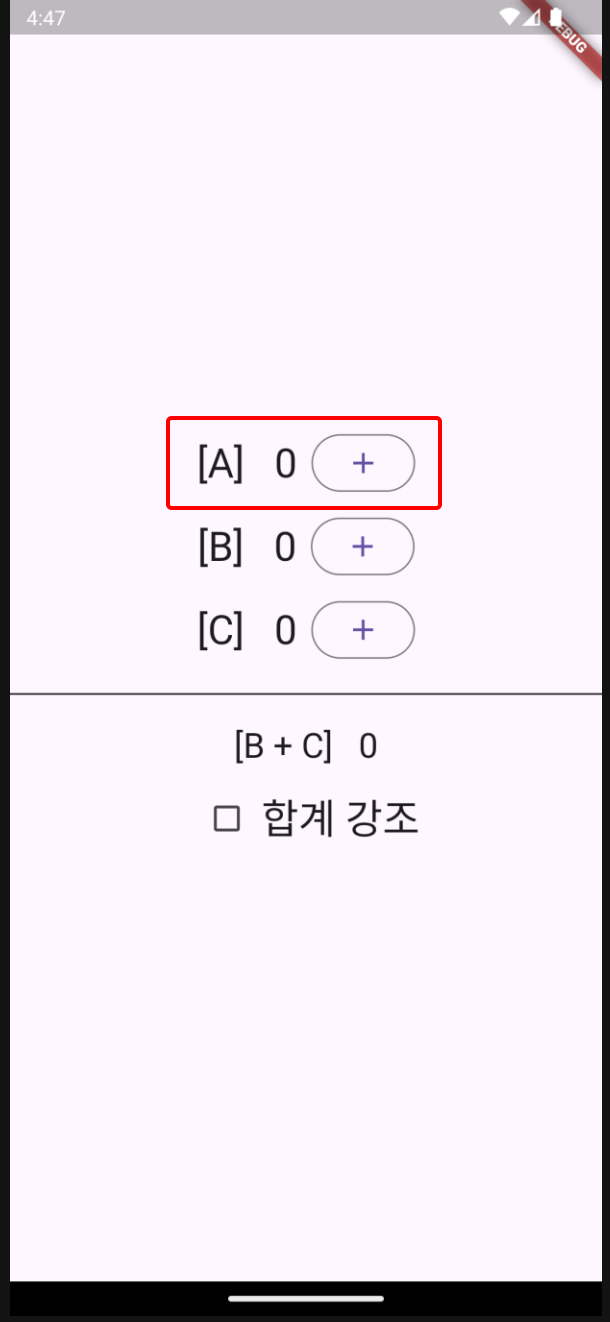
만약 `bloc` 파라미터가 생략되면 `BlocBuilder`는 `BlocProvider`와 현재 `BuildContext`를 사용하여 자동으로 bloc을 조회합니다.

```
BlocSelector<BlocA, BlocAState, SelectedState>(
  selector: (state) {
    // return selected state based on the provided state.
  },
  builder: (context, state) {
    // return widget here based on the selected state.
  },
);
```

https://bloclibrary.dev/flutter-bloc-concepts/#blocselector

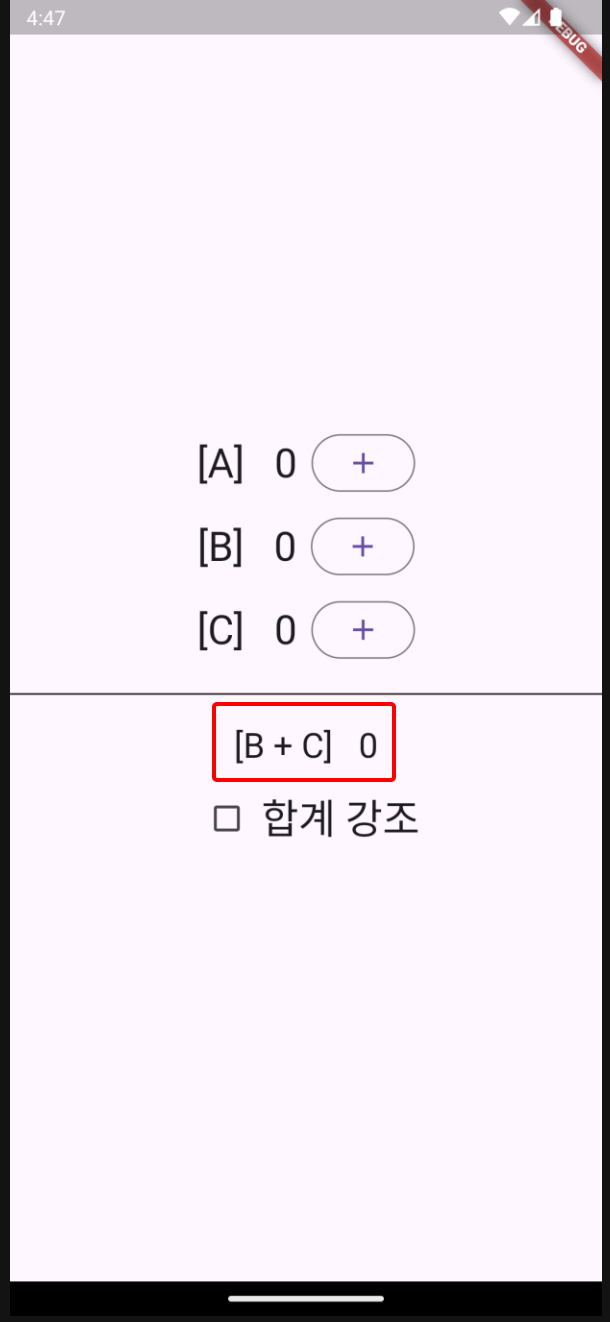



```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return CounterWidget(
      text: '[A]    ${state.counterA}',
      onPressed: context.read<SelectorNgCubit>().onTapIncrementA,
    );  // CounterWidget
  },
),  // BlocBuilder
```

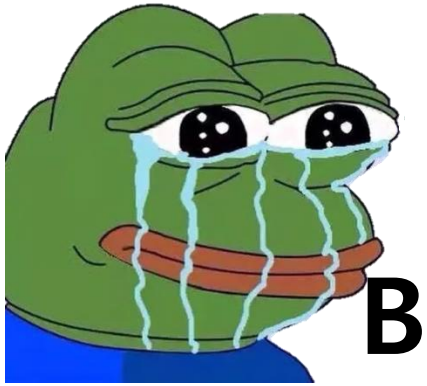



```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return CounterWidget(
      text: '[A]    ${state.counterA}',
      onPressed: context.read<SelectorNgCubit>().onTapIncrementA,
    );  // CounterWidget
  },
),  // BlocBuilder
```

```dart
BlocSelector<SelectorNgCubit, SelectorNgState, int>(
  selector: (state) => state.counterA,
  builder: (context, counterA) {
    return CounterWidget(
      text: '[A]    $counterA',
      onPressed: context.read<SelectorNgCubit>().onTapIncrementA,
    );  // CounterWidget
  },
),  // BlocSelector
```




```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```



값이 1개가 아니면

BlocSelector 적용이 어렵네요

`builder` 함수가 호출되는 시점을 세밀하게 제어하기 위해 선택적 `buildWhen` 파라미터가 제공됩니다. `buildWhen`은 이전 bloc state와 현재 bloc state를 가져온 후 boolean을 반환합니다. `buildWhen`이 true를 반환하면 `builder`가 `state`와 함께 호출되고 위젯이 다시 빌드됩니다. `buildWhen`이 false를 반환하면 `builder`는 `state`와 함께 호출되지 않으며 리빌드는 일어나지 않습니다.

```
BlocBuilder<BlocA, BlocAState>(
  buildWhen: (previousState, state) {
    // return true/false to determine whether or not
    // to rebuild the widget with state
  },
  builder: (context, state) {
    // return widget here based on BlocA's state
  },
);
```

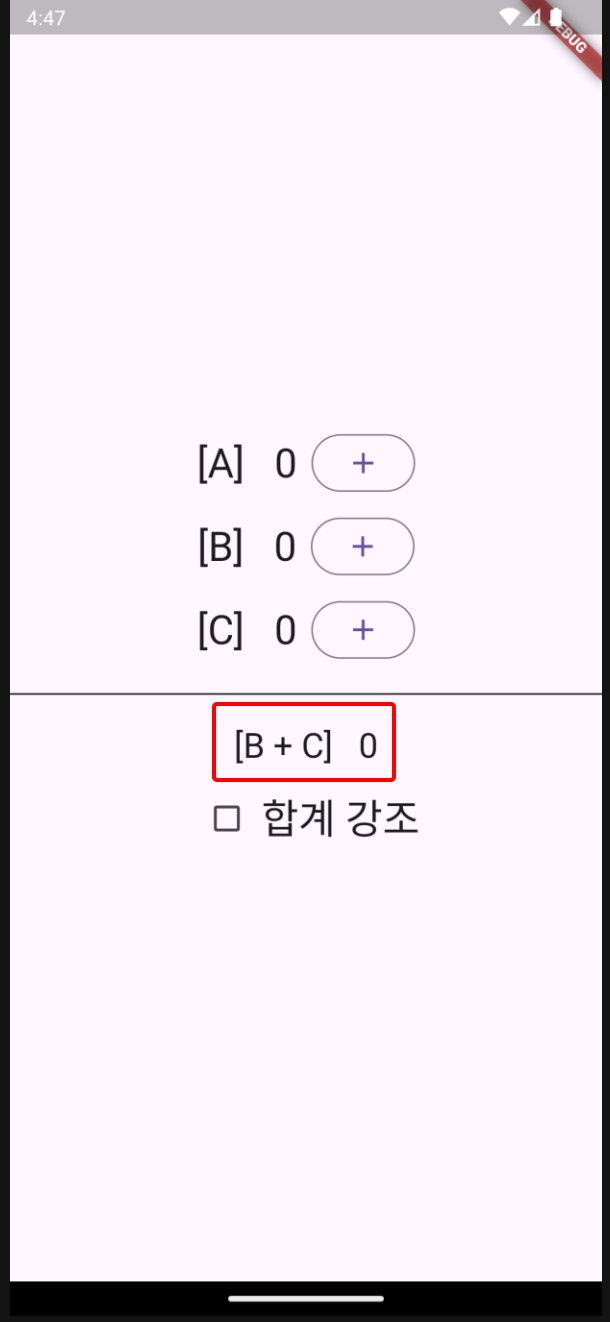
https://bloclibrary.dev/flutter-bloc-concepts/#blocbuilder

`builder` 함수가 호출되는 시점을 세밀하게 제어하기 위해 선택적 `buildWhen` 파라미터가 제공됩니다.

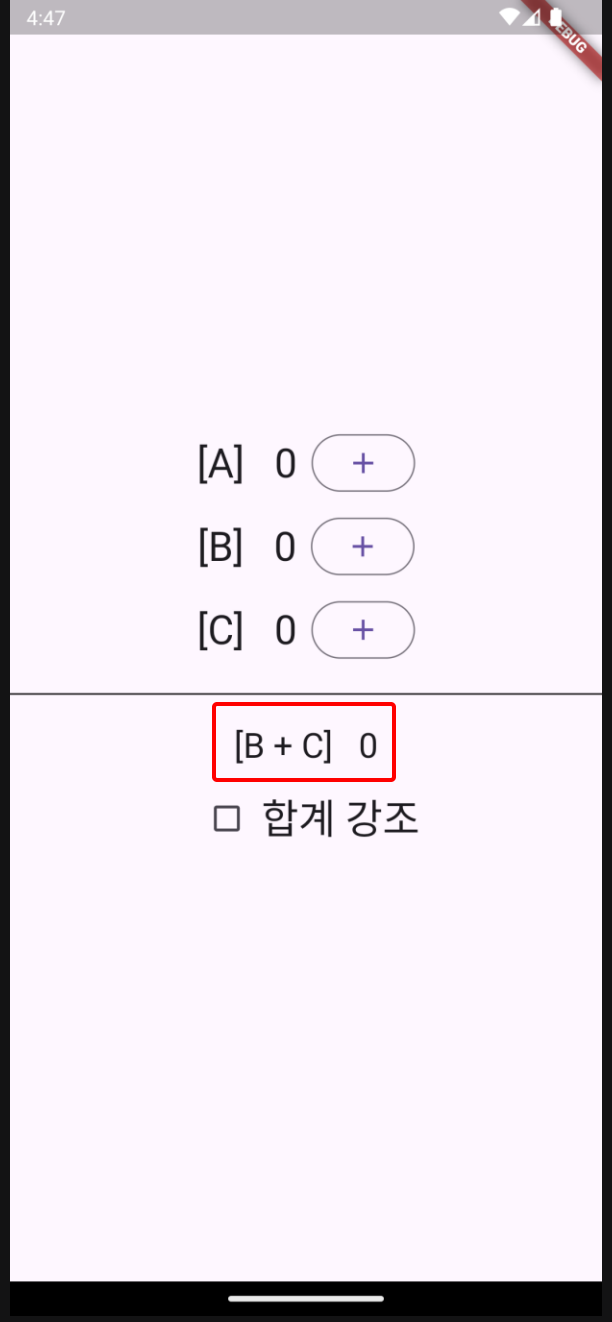
`buildWhen`은 이전 bloc state와 현재 bloc state를 가져온 후 boolean을 반환합니다. `buildWhen`이 true를 반환하면 `builder`가 `state`와 함께 호출되고 위젯이 다시 빌드됩니다. `buildWhen`이 false를 반환하면 `builder`는 `state`와 함께 호출되지 않으며 리빌드는 일어나지 않습니다.

```
BlocBuilder<BlocA, BlocAState>(
  buildWhen: (previousState, state) {
    // return true/false to determine whether or not
    // to rebuild the widget with state
  },
  builder: (context, state) {
    // return widget here based on BlocA's state
  },
);
```

https://bloclibrary.dev/flutter-bloc-concepts/#blocbuilder



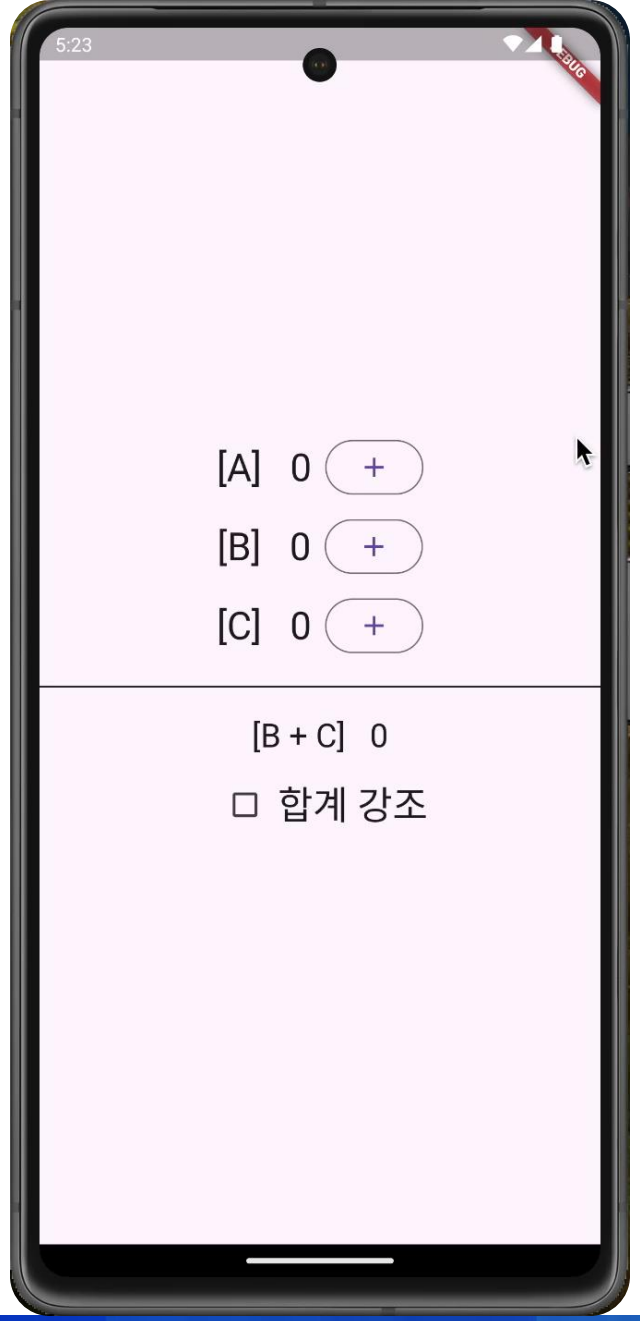

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```




```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

이제 잘 되겠지요? 🙏


5:23
DEBUG
[A] 0 +
[B] 0 +
[C] 0 +
[B + C] 0
합계 강조

아직 더 개선 가능합니다 !!!

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]   ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

유지보수가 어려운
중첩된 조건문

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

state 전체 접근이 가능해
bug 발생 가능성 내포

유지보수 어려운 조건문

State 노출로 bug 발생 가능성 내포

그래서

BlocSelector2 ... BlocSelector6

만들었습니다

```dart
class BlocSelector3<B extends StateStreamable<S>, S, T1, T2, T3> extends StatelessWidget {
  const BlocSelector3({
    super.key,
    this.bloc,
    required this.selector1,
    required this.selector2,
    required this.selector3,
    required this.builder,
  });

  final B? bloc;
  final BlocWidgetSelector<S, T1> selector1;
  final BlocWidgetSelector<S, T2> selector2;
  final BlocWidgetSelector<S, T3> selector3;
  final Function(BuildContext context, T1 state1, T2 state2, T3 state3) builder;

  @override
  Widget build(BuildContext context) {
    return BlocSelector<B, S, SampleData3<T1, T2, T3>>(
      bloc: bloc,
      selector: (state) {
        return SampleData3(
          selector1(state),
          selector2(state),
          selector3(state),
        );  // SampleData3
      },
      builder: (context, state) => builder(context, state.t1, state.t2, state.t3),
    );  // BlocSelector
  }
}
```

https://github.com/hohoins/flutter_bloc_abc_studio/blob/main/lib/common/bloc_selector.dart

```dart
class BlocSelector3<B extends StateStreamable<S>, S, T1, T2, T3> extends StatelessWidget {
  const BlocSelector3({
    super.key,
    this.bloc,
    required this.selector1,
    required this.selector2,
    required this.selector3,
    required this.builder,
  });

  final B? bloc;
  final BlocWidgetSelector<S, T1> selector1;
  final BlocWidgetSelector<S, T2> selector2;
  final BlocWidgetSelector<S, T3> selector3;
  final Function(BuildContext context, T1 state1, T2 state2, T3 state3) builder;

  @override
  Widget build(BuildContext context) {
    return BlocSelector<B, S, SampleData3<T1, T2, T3>>(
      bloc: bloc,
      selector: (state) {
        return SampleData3(
          selector1(state),
          selector2(state),
          selector3(state),
        );  // SampleData3
      },
      builder: (context, state) => builder(context, state.t1, state.t2, state.t3),
    );  // BlocSelector
  }
}
```

https://github.com/hohoins/flutter_bloc_abc_studio/blob/main/lib/common/bloc_selector.dart

BlocSelector3

적용해 봅시다

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]   ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]   ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

```dart
BlocSelector3<SelectorNgCubit, SelectorNgState, int, int, bool?>(
  selector1: (state) => state.counterB,
  selector2: (state) => state.counterC,
  selector3: (state) => state.isHighlight,
  builder: (context, counterB, counterC, isHighlight) {
    return ColorEffect(
      child: Text('[B + C]   ${counterB + counterC}',
          style: isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocSelector3
```

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]   ${state.counterB + state.counterC}',
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

유지보수가 어려운
중첩된 조건문

```dart
BlocSelector3<SelectorNgCubit, SelectorNgState, int, int, bool?>(
  selector1: (state) => state.counterB,
  selector2: (state) => state.counterC,
  selector3: (state) => state.isHighlight,
  builder: (context, counterB, counterC, isHighlight) {
    return ColorEffect(
      child: Text('[B + C]   ${counterB + counterC}',
          style: isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocSelector3
```

해결

```dart
BlocBuilder<SelectorNgCubit, SelectorNgState>(
  buildWhen: (previous, current) =>
      previous.counterB != current.counterB ||
      previous.counterC != current.counterC ||
      previous.isHighlight != current.isHighlight,
  builder: (context, state) {
    return ColorEffect(
      child: Text('[B + C]    ${state.counterB + state.counterC
          style: state.isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocBuilder
```

state 전체 접근이 가능해
bug 발생 가능성 내포

```dart
BlocSelector3<SelectorNgCubit, SelectorNgState, int, int, bool?>(
  selector1: (state) => state.counterB,
  selector2: (state) => state.counterC,
  selector3: (state) => state.isHighlight,
  builder: (context, counterB, counterC, isHighlight) {
    return ColorEffect(
      child: Text('[B + C]    ${counterB + counterC}',
          style: isHighlight == true
              ? Theme.of(context).textTheme.headlineLarge
              : Theme.of(context).textTheme.headlineSmall),  // Text
    );  // ColorEffect
  },
),  // BlocSelector3
```

해결

~~유지보수 어려운 조건문~~ 해결

~~State 노출로 bug 발생 가능성 내포~~ 해결

아무리 그래도...

2 ~ 6은 조금 어색한데요

BlocSelector2

BlocSelector3

BlocSelector4

BlocSelector5

BlocSelector6

provider 6.1.2

Published 6 months ago • dash-overflow.net Dart 3 compatible

SDK | FLUTTER

PLATFORM | ANDROID | IOS | LINUX | MACOS | WEB | WINDOWS

10.1K

Selector
Selector0
Selector2
Selector3
Selector4
Selector5
Selector6

Consumer
Consumer2
Consumer3
Consumer4
Consumer5
Consumer6

ProxyProvider0
ProxyProvider2
ProxyProvider3
ProxyProvider4
ProxyProvider5
ProxyProvider6

ProviderBuilder
ProxyProviderBuilder
ProxyProviderBuilder2
ProxyProviderBuilder3
ProxyProviderBuilder4
ProxyProviderBuilder5
ProxyProviderBuilder6

https://pub.dev/documentation/provider/latest/provider/provider-library.html

widget build 최적화 완료!

마무리로 테스트 코드 작성~

5. Bloc test 개선하기

Bloc은 세 가지 핵심 가치를 염두에 두고 설계되었습니다.

- **간단함:** 이해하기 쉽고 다양한 기술 수준의 개발자가 사용할 수 있음.
- **강력함:** 더 작은 구성요소로 구성하여 놀랍고 복잡한 애플리케이션을 만드는 데 도움이 됩니다.
- **테스트 가능:** 애플리케이션의 모든 측면을 쉽게 테스트하여 확신을 가지고 반복할 수 있습니다.

https://pub.dev/packages/bloc_test

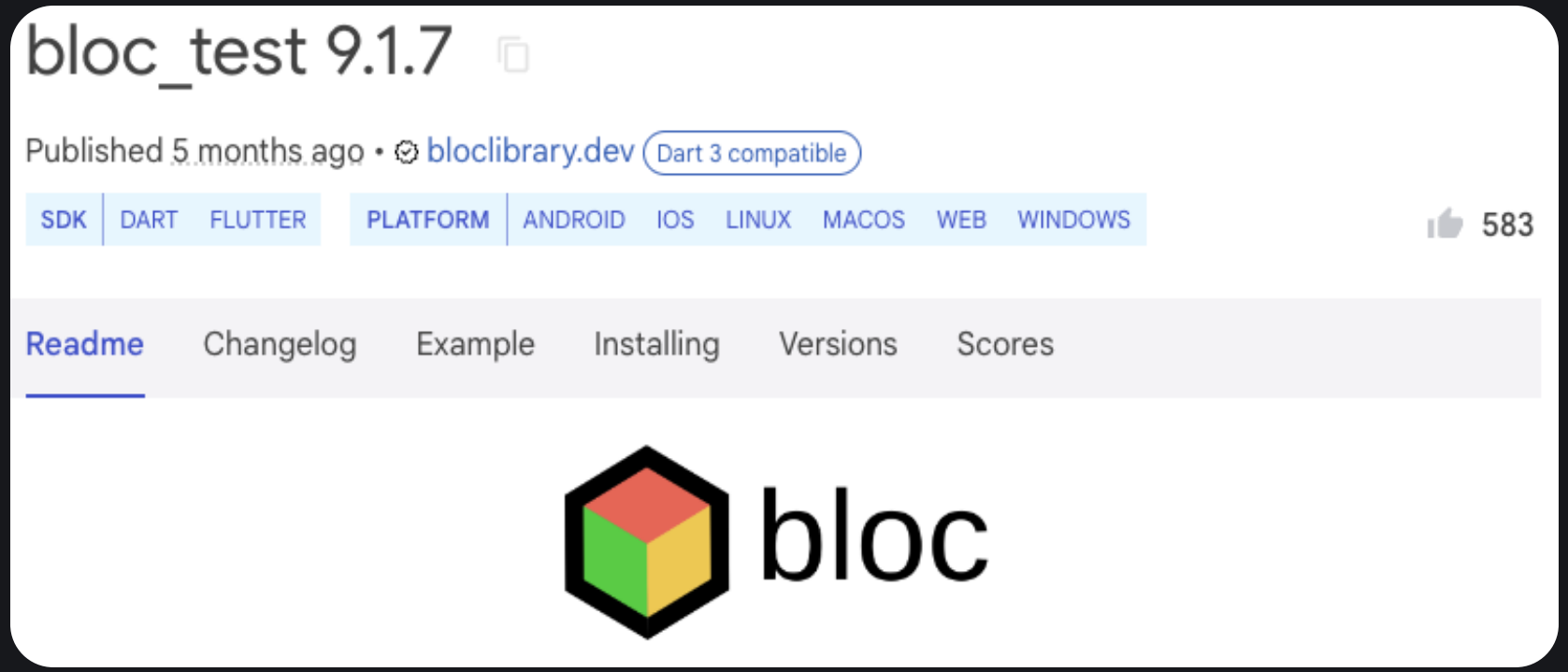
Bloc은 세 가지 핵심 가치를 염두에 두고 설계되었습니다.

- **간단함:** 이해하기 쉽고 다양한 기술 수준의 개발자가 사용할 수 있음.
- **강력함:** 더 작은 구성요소로 구성하여 놀랍고 복잡한 애플리케이션을 만드는 데 도움이 됩니다.
- **테스트 가능:** 애플리케이션의 모든 측면을 쉽게 테스트하여 확신을 가지고 반복할 수 있습니다.

https://pub.dev/packages/bloc_test

Bloc은 세 가지 핵심 가치를 염두에 두고 설계되었습니다.

- **간단함:** 이해하기 쉽고 다양한 기술 수준의 개발자가 사용할 수 있음.
- **강력함:** 더 작은 구성요소로 구성하여 놀랍고 복잡한 애플리케이션을 만드는 데 도움이 됩니다.
- **테스트 가능:** 애플리케이션의 모든 측면을 쉽게 테스트하여 확신을 가지고 반복할 수 있습니다.

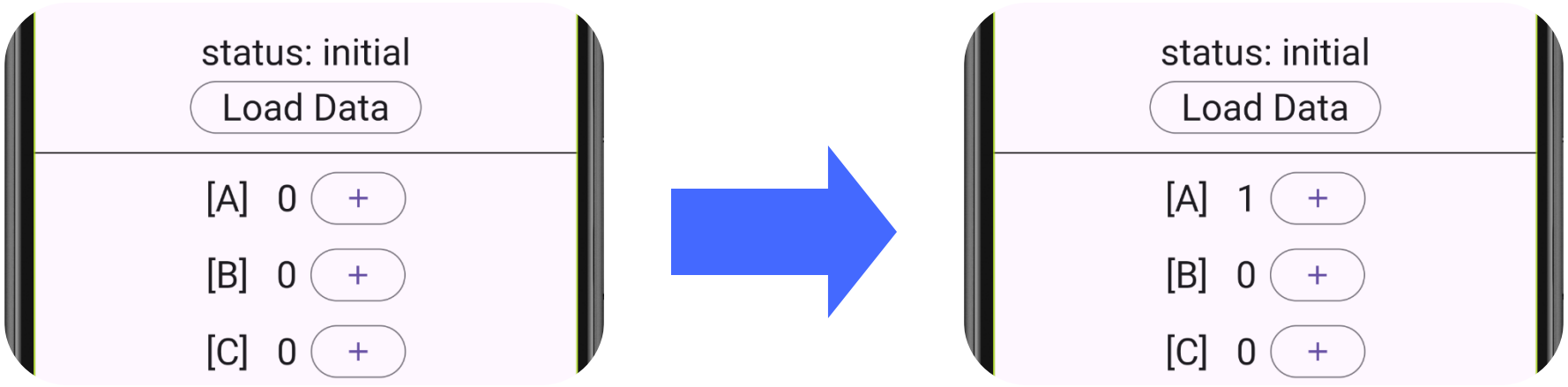


https://pub.dev/packages/bloc_test

```
blocTest(
  build: () => MyBloc(),
  act: (bloc) => bloc.add(MyEvent()),
  expect: () => [isA<MyState>()],
);
```

```
blocTest(
  build: () => MyBloc(),
  act: (bloc) => bloc.add(MyEvent()),
  expect: () => [isA<MyState>()],
);
```

```
blocTest(
  build: () => MyBloc(),
  act: (bloc) => bloc.add(MyEvent()),
  expect: () => [isA<MyState>()],
);
```


status: initial
Load Data
[A] 0 +
[B] 0 +
[C] 0 +
status: initial
Load Data
[A] 1 +
[B] 0 +
[C] 0 +

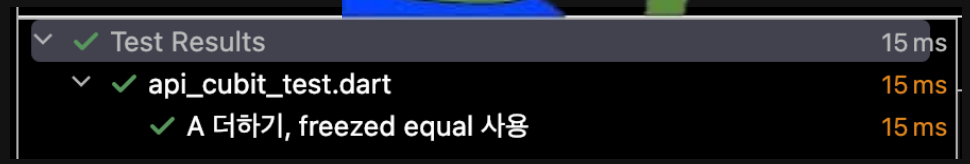
더하기 테스트 해봅시다

```dart
blocTest(
  'A 더하기, freezed equal 사용',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapIncrementA();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.initial,
      counterA: 1,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    )
  ],
);
```

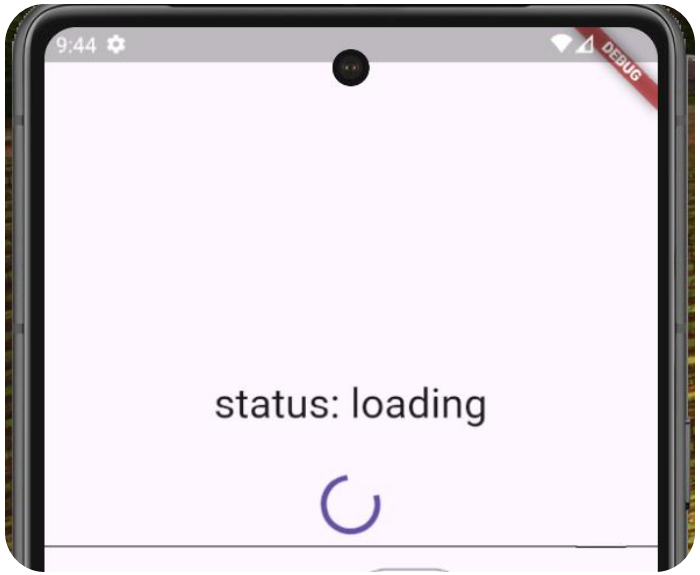
```dart
blocTest(
  'A 더하기, freezed equal 사용',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapIncrementA();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.initial,
      counterA: 1,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    )
  ],
);
```

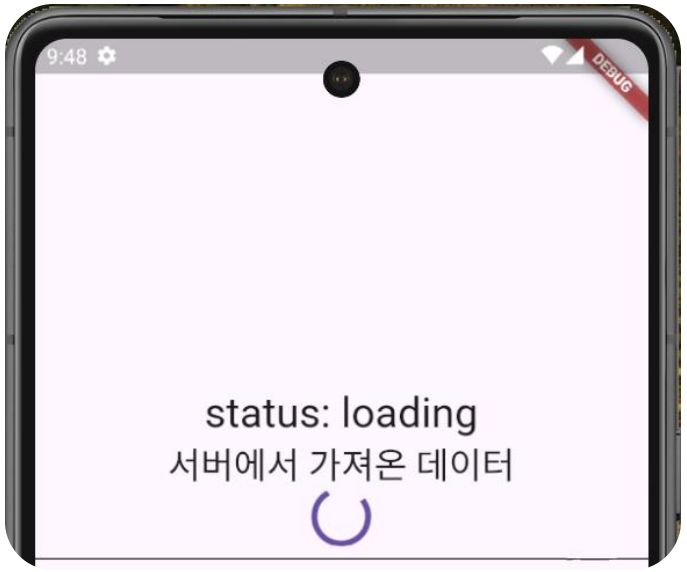
```dart
blocTest(
  'A 더하기, freezed equal 사용',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapIncrementA();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.initial,
      counterA: 1,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    )
  ],
);
```

```dart
blocTest(
  'A 더하기, freezed equal 사용',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapIncrementA();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.initial,
      counterA: 1,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    )
  ],
);
```



API 호출 테스트





emit 1

loading

emit 2

서버 데이터 처리

emit 3

success

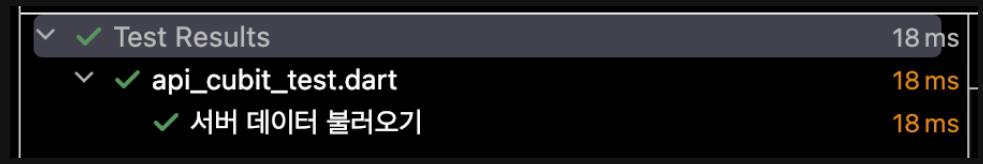
```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [



  ],
);
```

```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.success,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    )
  ],
);
```

```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.success,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    )
  ],
);
```

```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.success,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    )
  ],
);
```

```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.success,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    )
  ],
);
```



테스트 성공!!!

테스트 코드
작성이 어려운
문제가 있네요

```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.success,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    )
  ],
);
```

state 유지가 안 되는 문제
수작업으로 처리

```dart
enum ApiStatus { initial, loading, success }

@freezed
class ApiState with _$ApiState {
  const factory ApiState({
    required ApiStatus status,
    required String serverData,
    required int counterA,
    required int counterB,
    required int counterC,
    required bool? isHighlight,
  }) = _ApiState;
}
```

state 수정
ex) 멤버 변수명 변경

```dart
enum ApiStatus { initial, loading, success }

@freezed
class ApiState with _$ApiState {
  const factory ApiState({
    required ApiStatus status,
    required String serverData,
    required int counterA,
    required int counterB,
    required int counterC,
    required bool? isHighlight,
  }) = _ApiState;
}
```

수정할 내용이 많습니다
state 수정 불리한 문제

```dart
blocTest(
  '서버 데이터 불러오기',
  build: () => ApiCubit(),
  act: (cubit) {
    cubit.onTapLoadData();
  },
  expect: () => [
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.loading,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    ),
    const ApiState(
      status: ApiStatus.success,
      counterA: 0,
      counterB: 0,
      counterC: 0,
      serverData: '서버에서 가져온 데이터',
      isHighlight: false,
    )
  ],
);
```

state 유지가 안 되는 문제

state 수정 불리한 문제

해결해 봅시다

```dart
class StateKeeper<T> {
  StateKeeper(this.state);

  T state;

  T emit(T newState) {
    state = newState;
    return state;
  }
}
```

```dart
class StateKeeper<T> {
  StateKeeper(this.state);

  T state;

  T emit(T newState) {
    state = newState;
    return state;
  }
}
```

```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

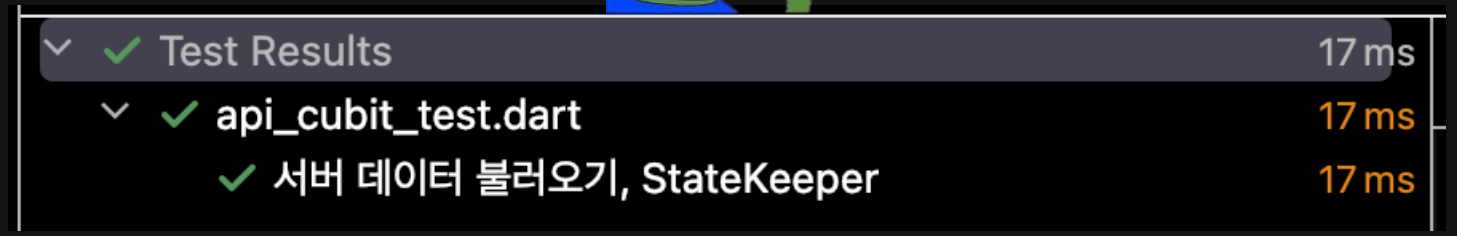
```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```



```dart
late StateKeeper<ApiState> keep;
blocTest(
  '서버 데이터 불러오기, StateKeeper',
  build: () => ApiCubit(),
  act: (cubit) {
    keep = StateKeeper(cubit.state);
    cubit.onTapLoadData();
  },
  expect: () => [
    keep.emit(keep.state.copyWith(status: ApiStatus.loading)),
    keep.emit(keep.state.copyWith(serverData: '서버에서 가져온 데이터')),
    keep.emit(keep.state.copyWith(status: ApiStatus.success)),
  ],
);
```

테스트할 값에만 집중 가능
state class 변화 대응이 좋음

~~tate 유지가 안 되는 문제~~ 해결

~~state 수정 불리한 문제~~ 해결


소개해 드린 모든 내용은
데마에칸 MerchantApp
제품 개발에 적용된 내용입니다
도움이 되었으면 좋겠습니다
Future<Flutter> 2024

Bloc

야무지게

사용하시기를 바랍니다

감사합니다

Flutter Bloc을 제품 개발에 야무지게 적용하기

샘플 코드 깃허브

https://github.com/hohoins/flutter_bloc_abc_studio