Google Developers

Experts

Unidirectional Data Flow Rocks!

Introducing Redux, Cubits and Blocs.

Jay (Jeroen Meijer)
Flutter & Dart GDE
@jfkdev

State Management

State Management

State Management

- Using built-in solutions is fine.

State Management

- Using built-in solutions is fine.
- Keep state local, pass it around.

State Management

- Using built-in solutions is fine.
- Keep state local, pass it around.
- Becomes a problem when the app and scope grows.

State Management

- Using built-in solutions is fine.
- Keep state local, pass it around.
- Becomes a problem when the app and scope grows.
- We need some state management design pattern.

State Management

State Management

- Unidirectional data flow.

State Management

- Unidirectional data flow.
- A design pattern for state management.

State Management

- Unidirectional data flow.
- A design pattern for state management.
- Makes the flow of data structured, consistent, predictable and testable.

State Management

- Unidirectional data flow.
- A design pattern for state management.
- Makes the flow of data structured, consistent, predictable and testable.
- Helps separate views from business logic.

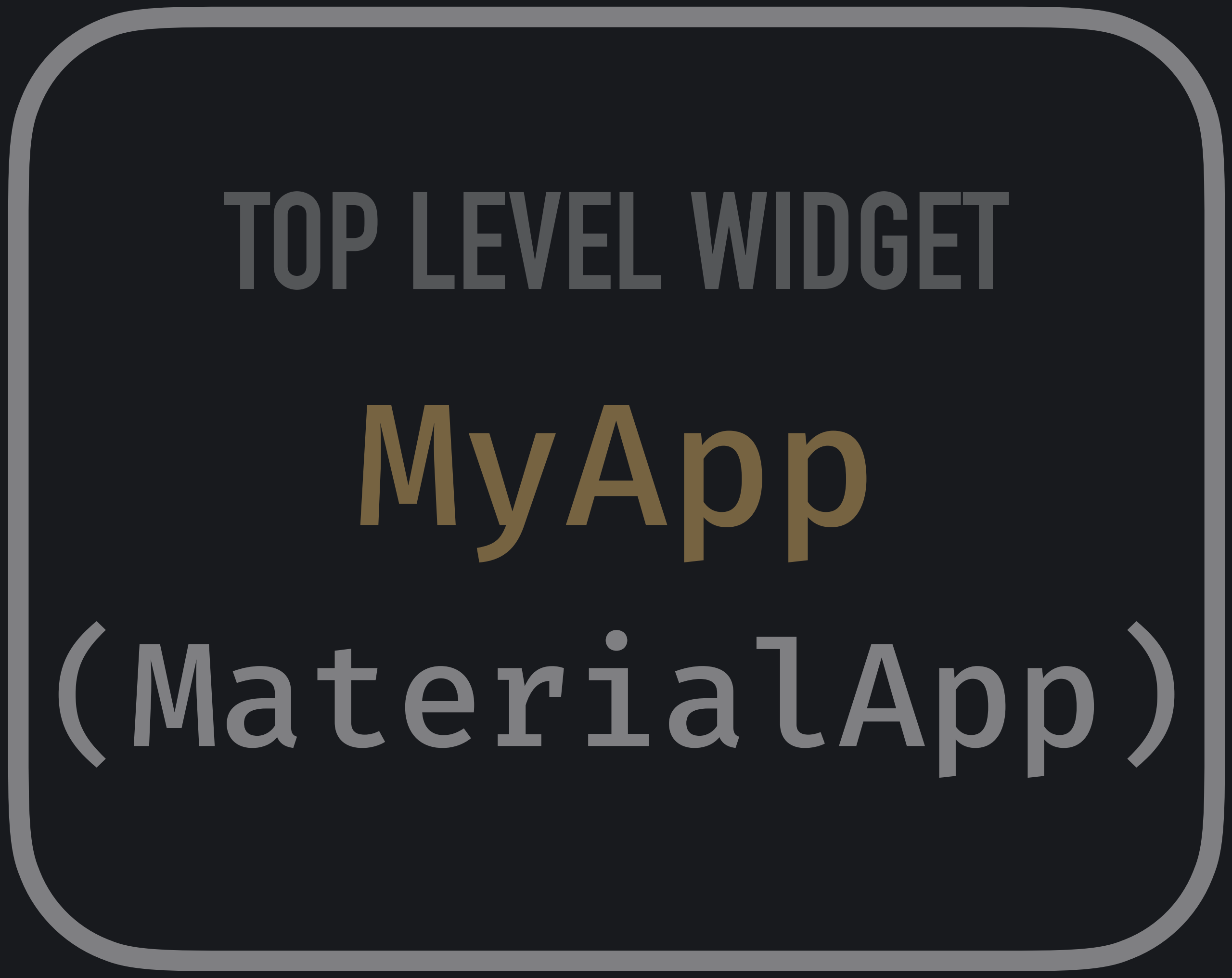
State Management

- Unidirectional data flow.
- A design pattern for state management.
- Makes the flow of data structured, consistent, predictable and testable.
- Helps separate views from business logic.
- Works well with declarative programming.

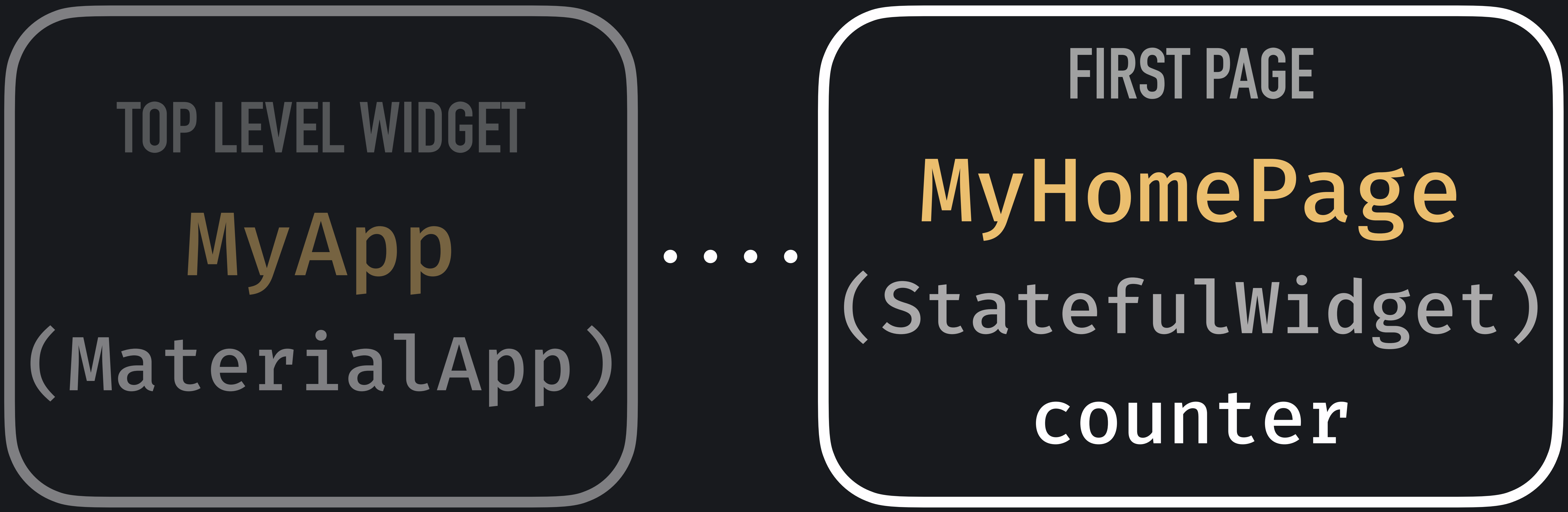
Example

Example

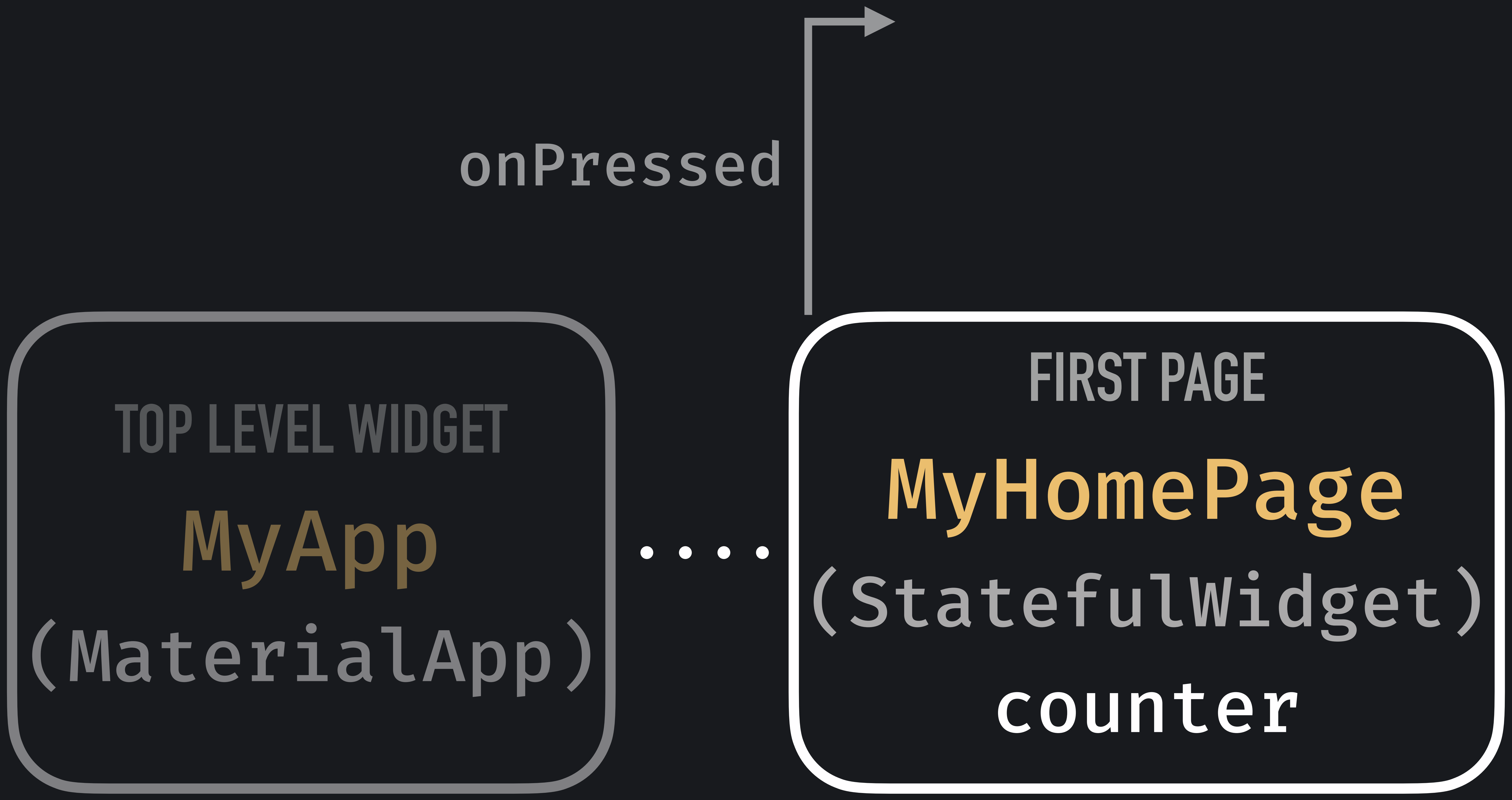
Example



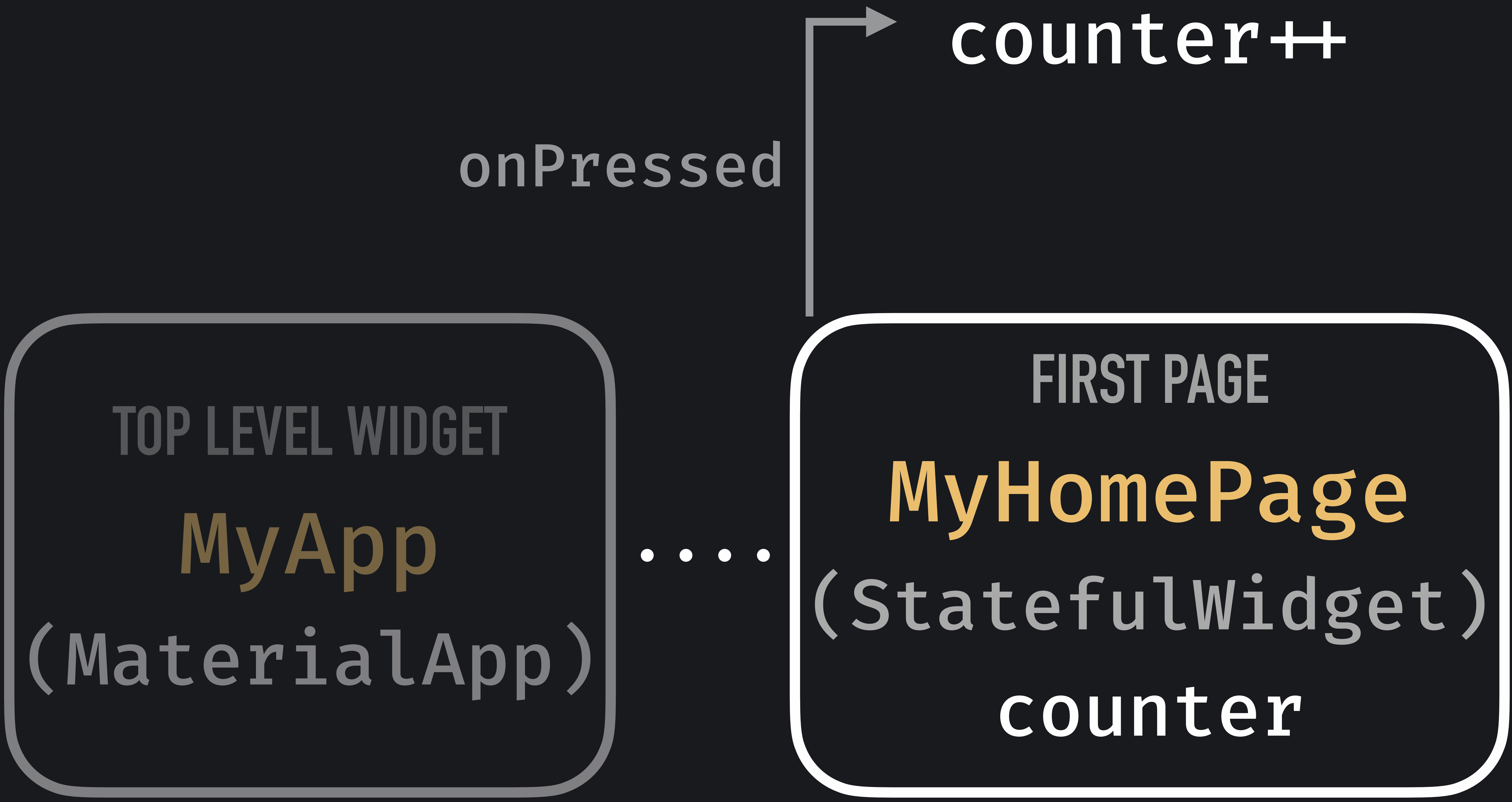
Example


TOP LEVEL WIDGET
MyApp
(MaterialApp)
FIRST PAGE
MyHomePage
(StatefulWidget)
counter

Example

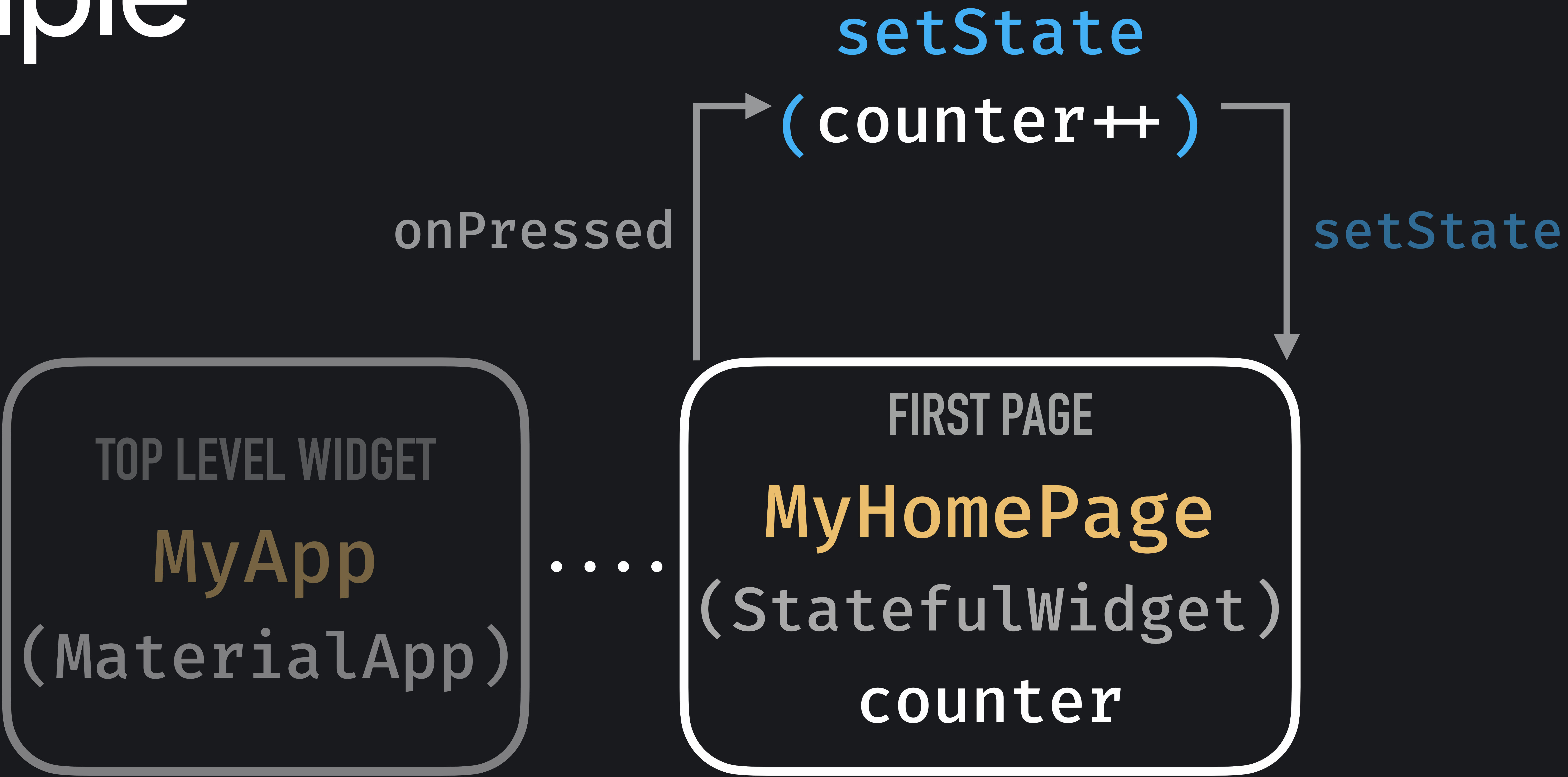

onPressed
TOP LEVEL WIDGET
MyApp
(MaterialApp)
FIRST PAGE
MyHomePage
(StatefulWidget)
counter

Example



counter++
onPressed
TOP LEVEL WIDGET
MyApp
(MaterialApp)
FIRST PAGE
MyHomePage
(StatefulWidget)
counter

Example


setState
(counter++)
onPressed
setState
TOP LEVEL WIDGET
MyApp
(MaterialApp)
FIRST PAGE
MyHomePage
(StatefulWidget)
counter

SOME CODE

SOME CODE

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: new MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

```
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _MyHomePageState createState() => new _MyHomePageState();
}
```

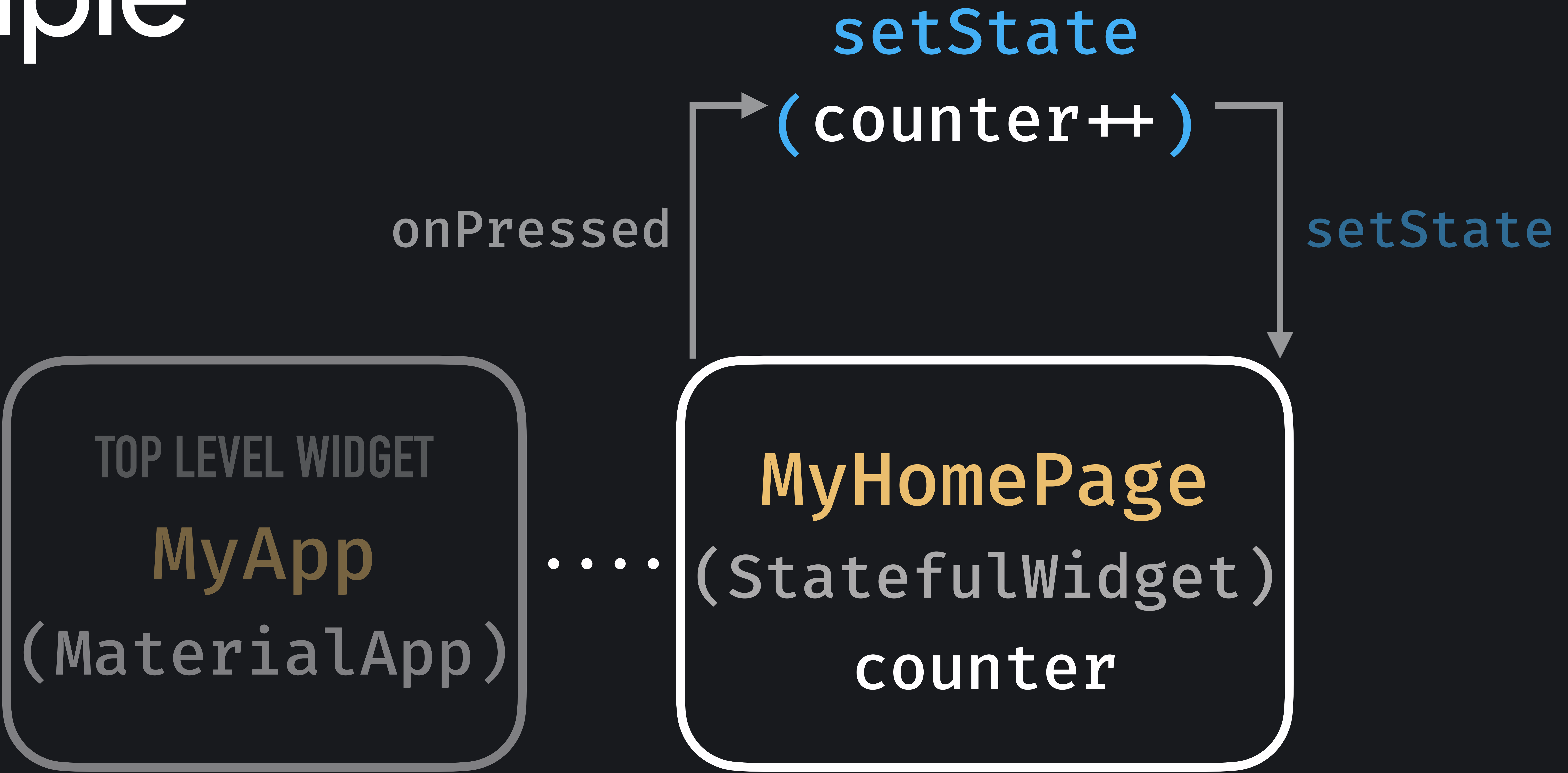
```
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }
  ...
    floatingActionButton: new FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
  ...
```

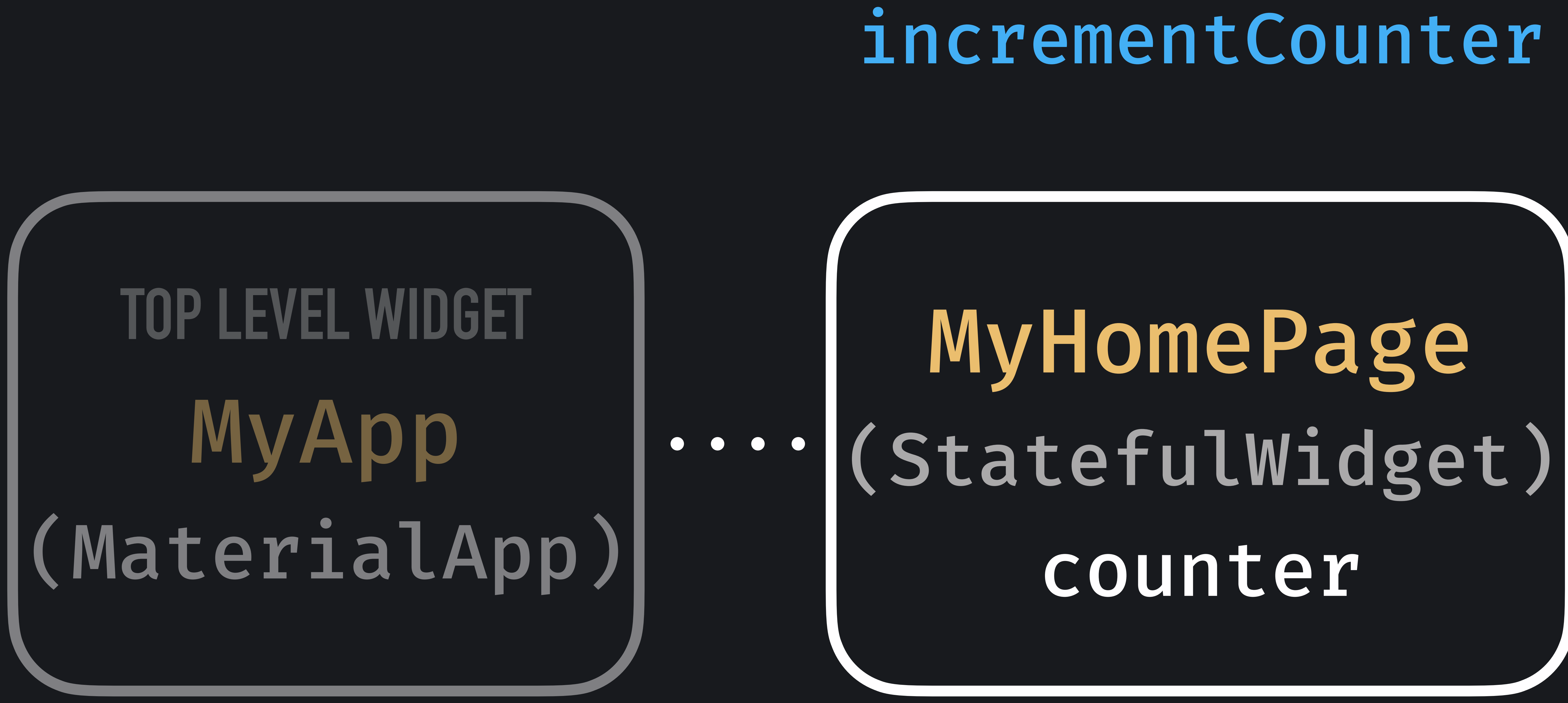
```
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }
  ...
    floatingActionButton: new FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
  ...
```

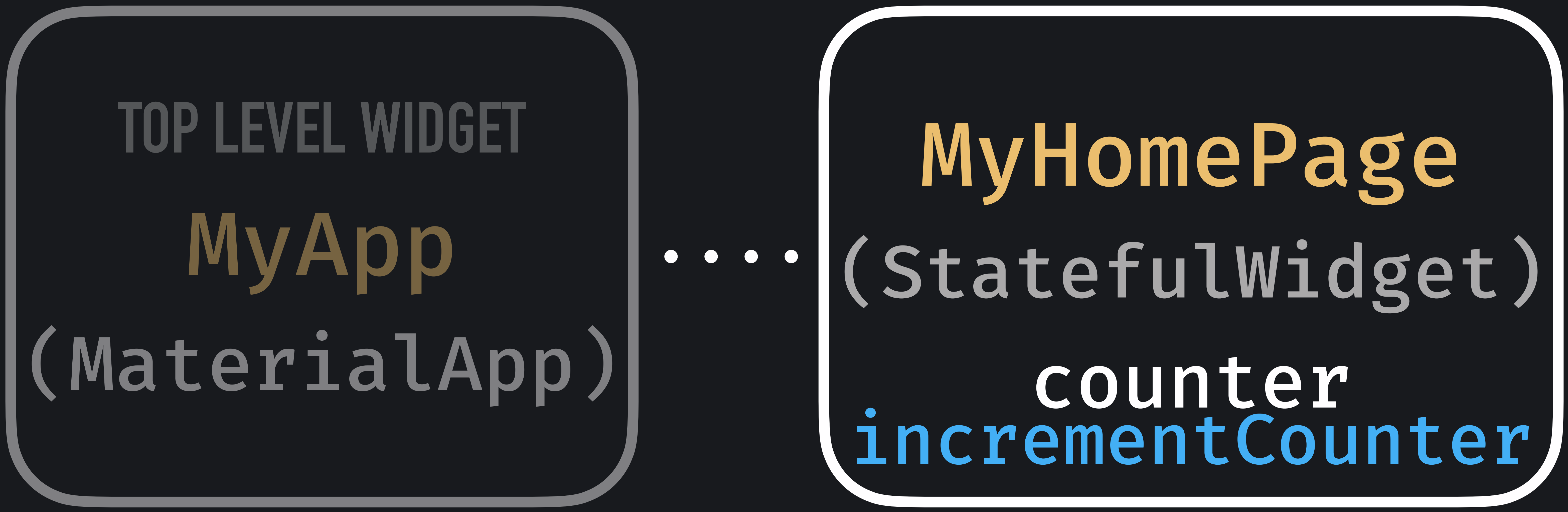
Example


setState
(counter++)
onPressed
setState
TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter

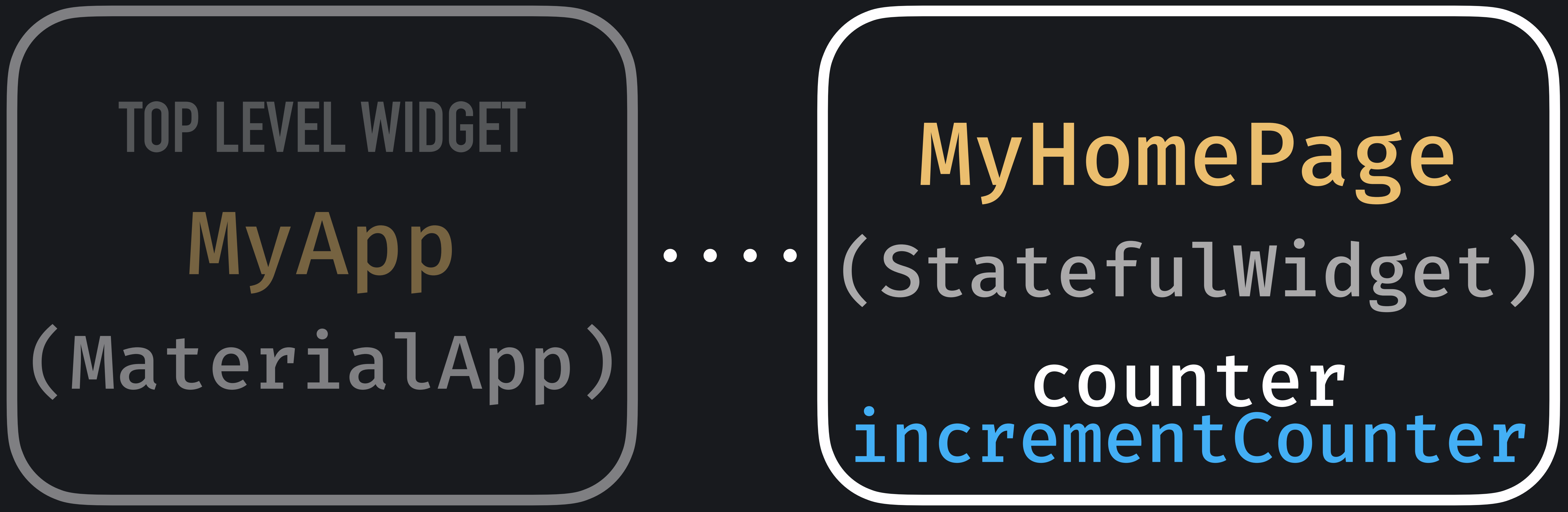
Example


incrementCounter
TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter

Example


TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter

Example


TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter

Example

TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)

Example

TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

Example

TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

Example

TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

```
new MySecondPage(
  counter: _counter,
  incrementCounter: _incrementCounter,
)
```

Example

TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

```
new MySecondPage(
  counter: _counter,
  incrementCounter: _incrementCounter,
)
```

Example

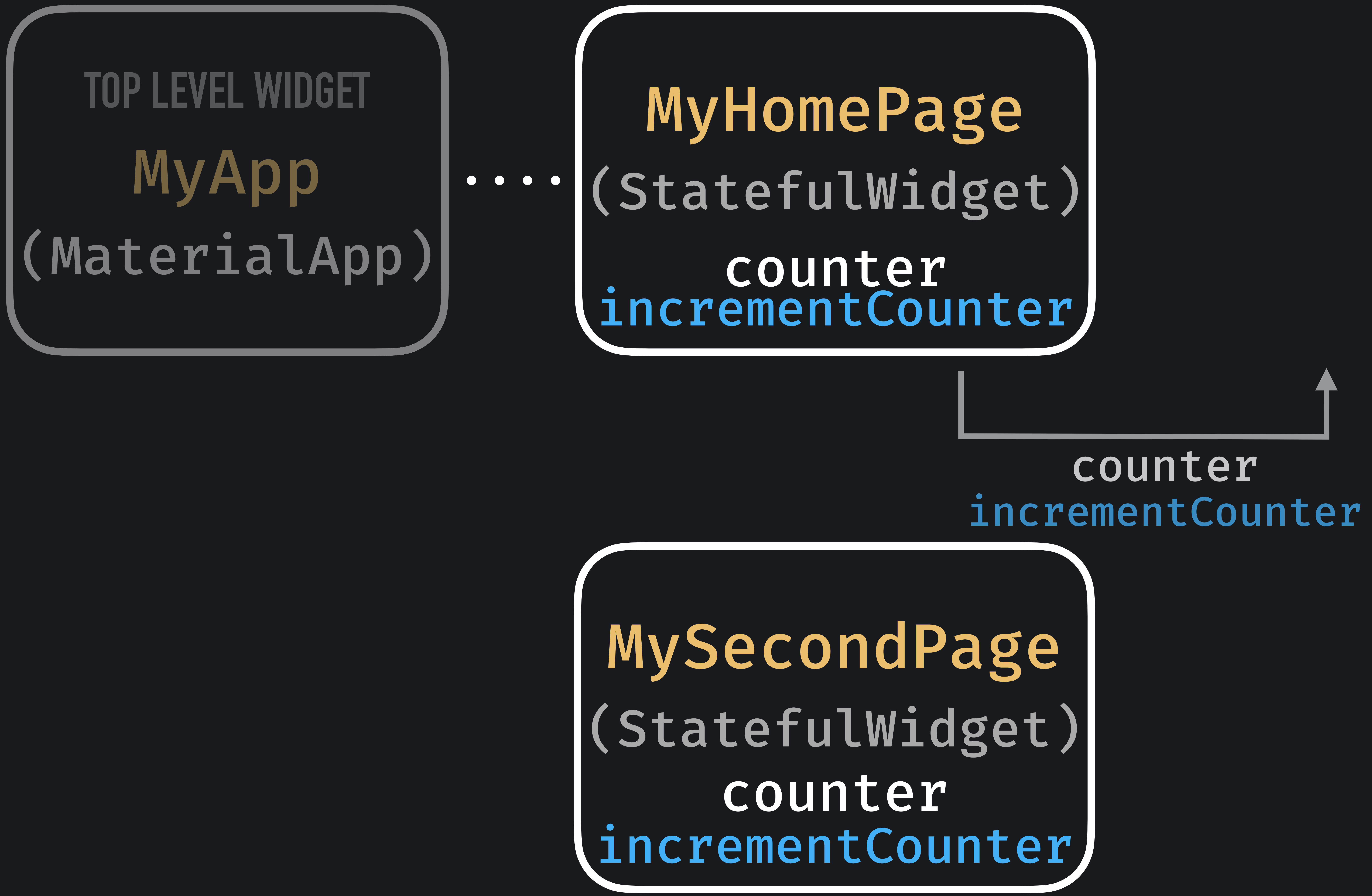
TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

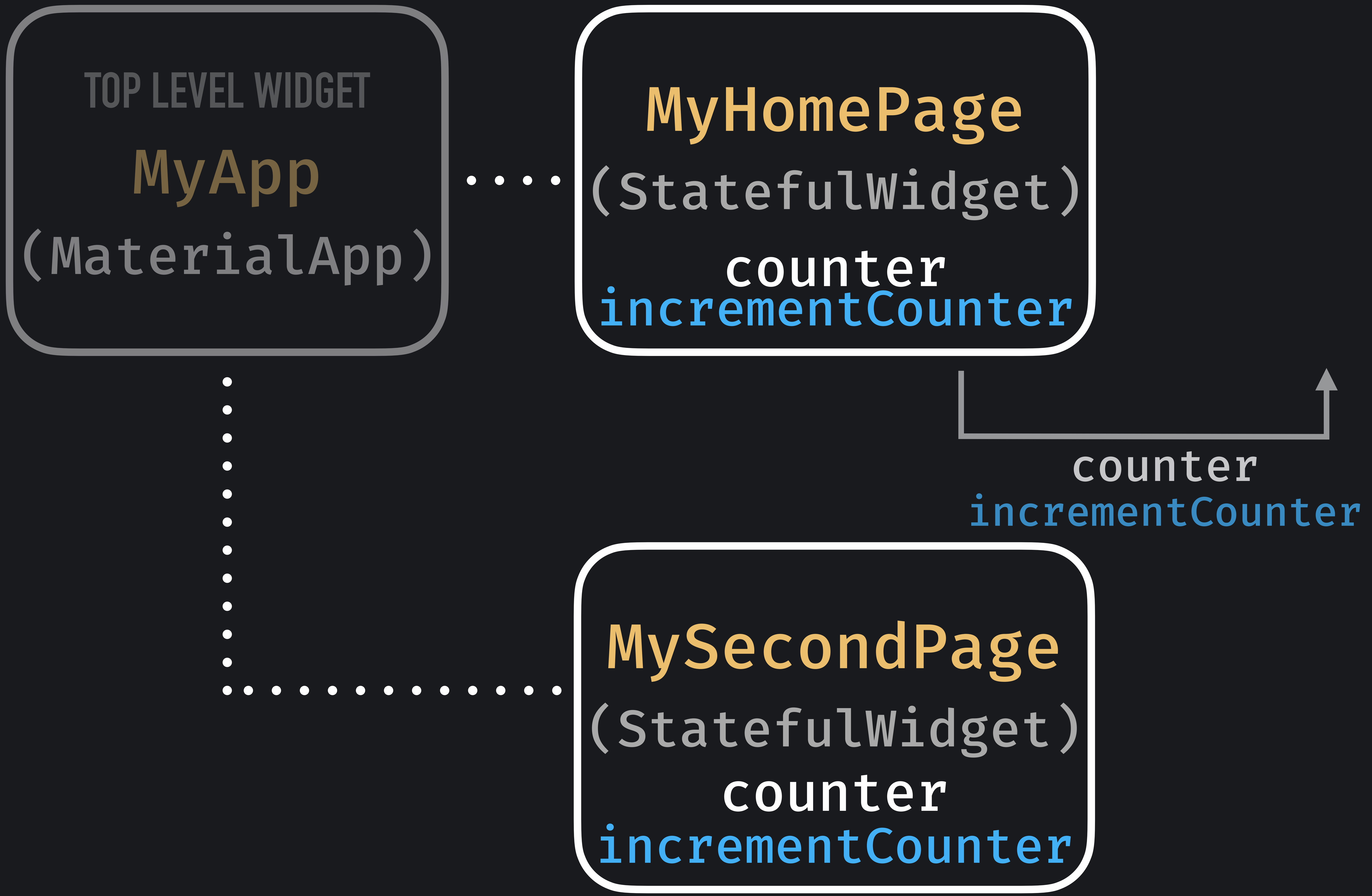
Example

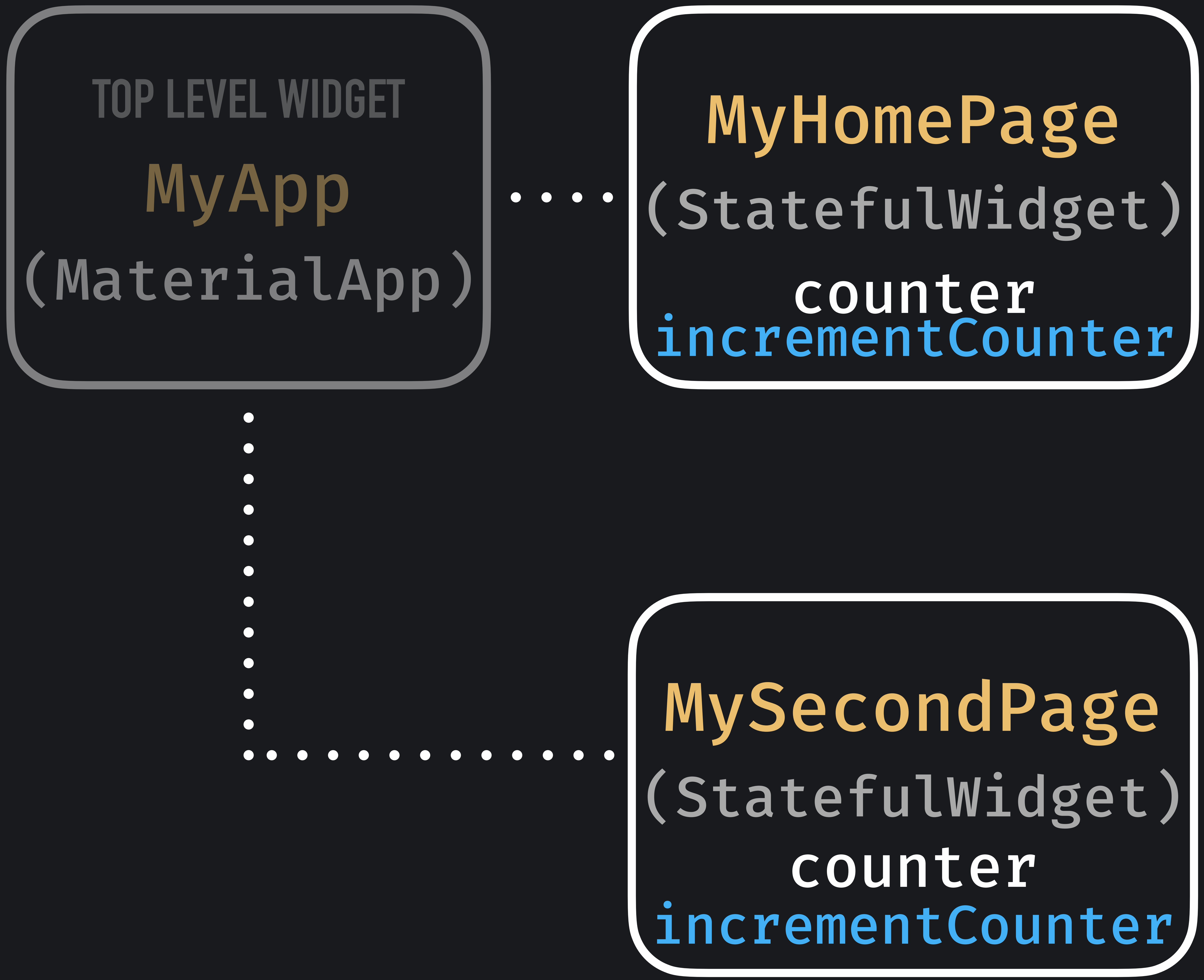
TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

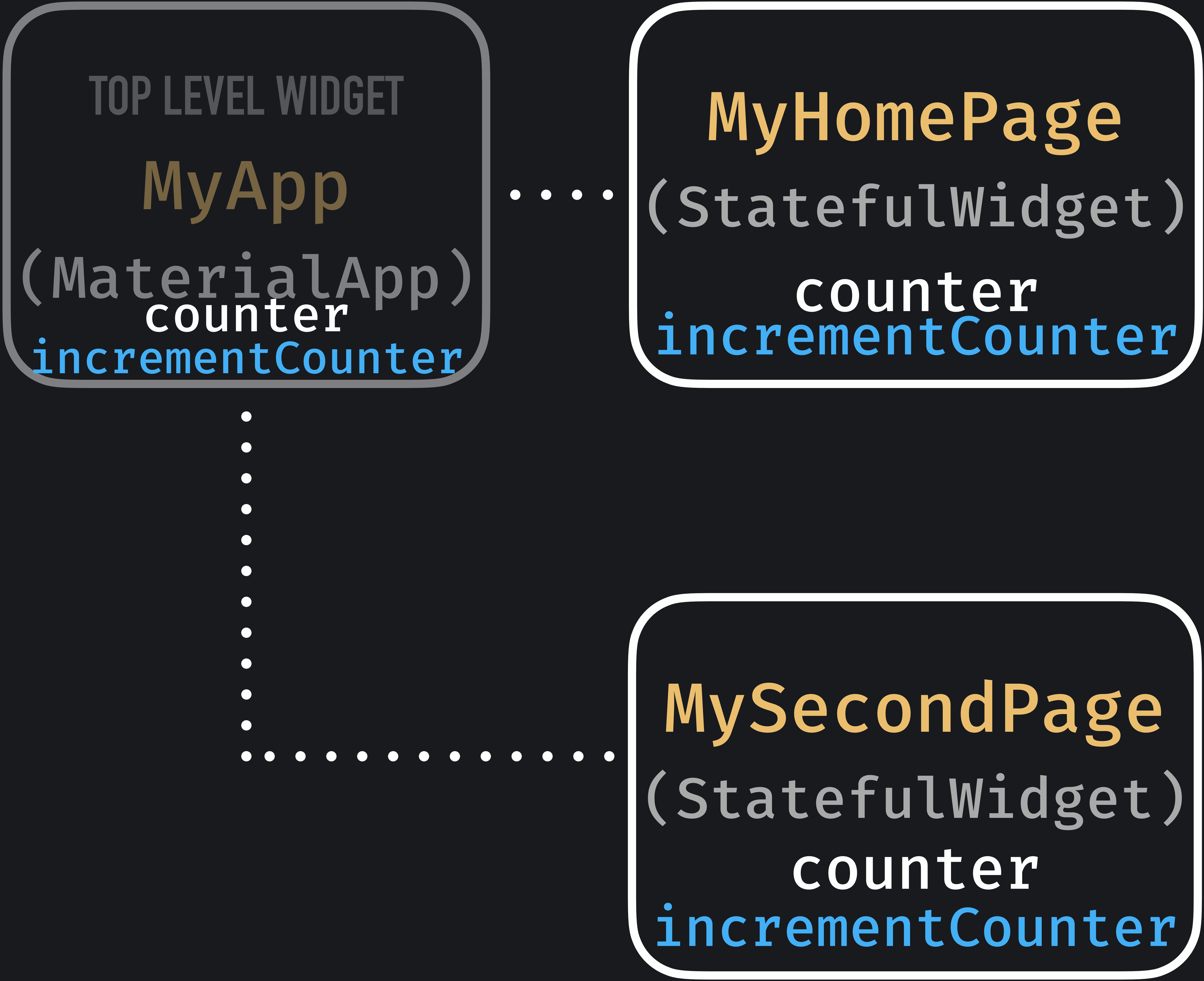
Example

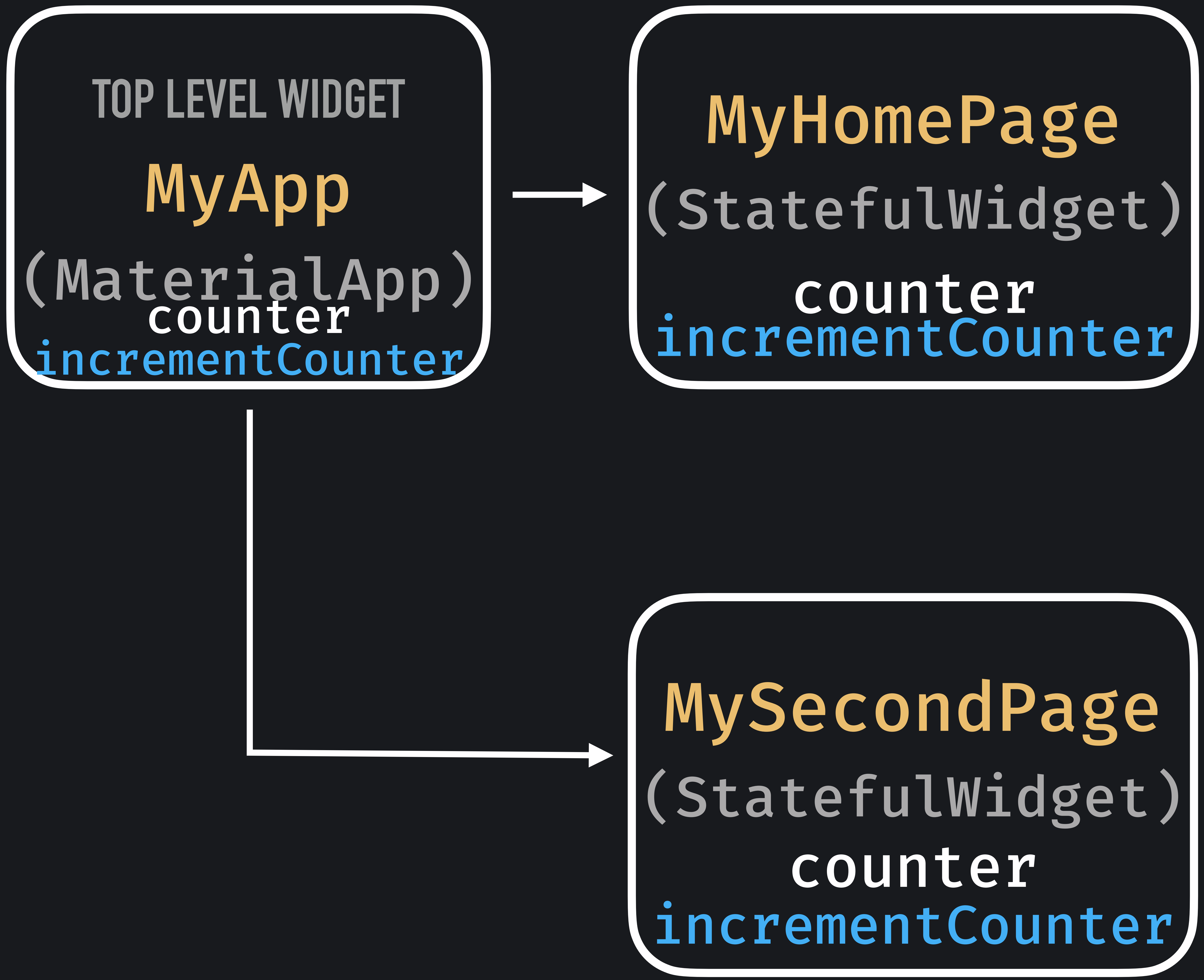
TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter
counter
incrementCounter


TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter


TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter


TOP LEVEL WIDGET
MyApp
(MaterialApp)
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter


TOP LEVEL WIDGET
MyApp
(MaterialApp)
counter
incrementCounter
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter


TOP LEVEL WIDGET
MyApp
(MaterialApp)
counter
incrementCounter
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter

TOP LEVEL WIDGET
MyApp
(MaterialApp)
counter
incrementCounter
MyHomePage
(StatefulWidget)
counter
incrementCounter
MySecondPage
(StatefulWidget)
counter
incrementCounter
AnotherPage
AnotherPage
AnotherPage
AnotherPage
AnotherPage
This is
Really?
Stop

Unidirectional Data Flow

Unidirectional Data Flow

Unidirectional Data Flow

- An immutable **state** that represents (a part of) your app.

Unidirectional Data Flow

- An immutable **state** that represents (a part of) your app.

- A collection of **events** that represent actions.

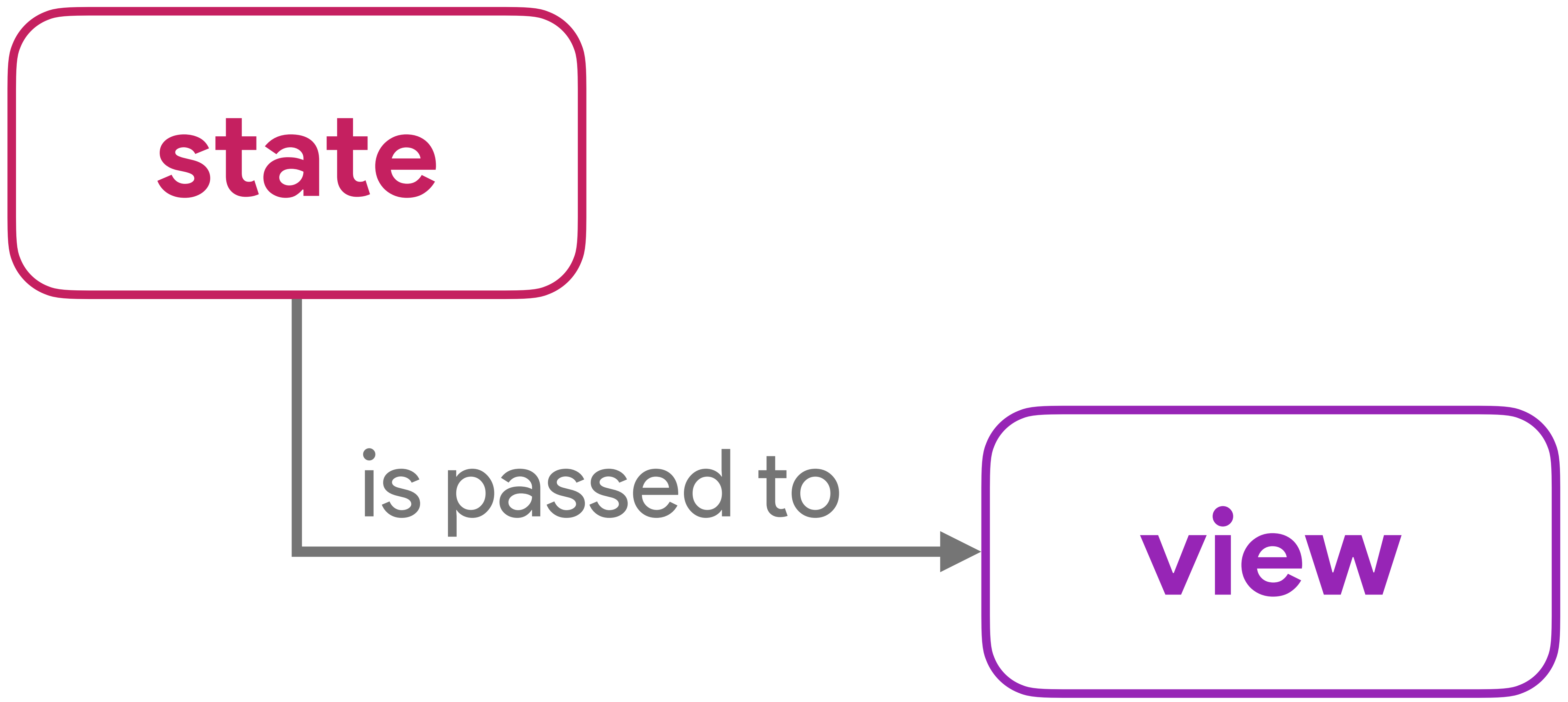
Unidirectional Data Flow

- An immutable **state** that represents (a part of) your app.
- A collection of **events** that represent actions.
- A **reducer/pure function** that takes a **state** and an **event** and produces a new state.

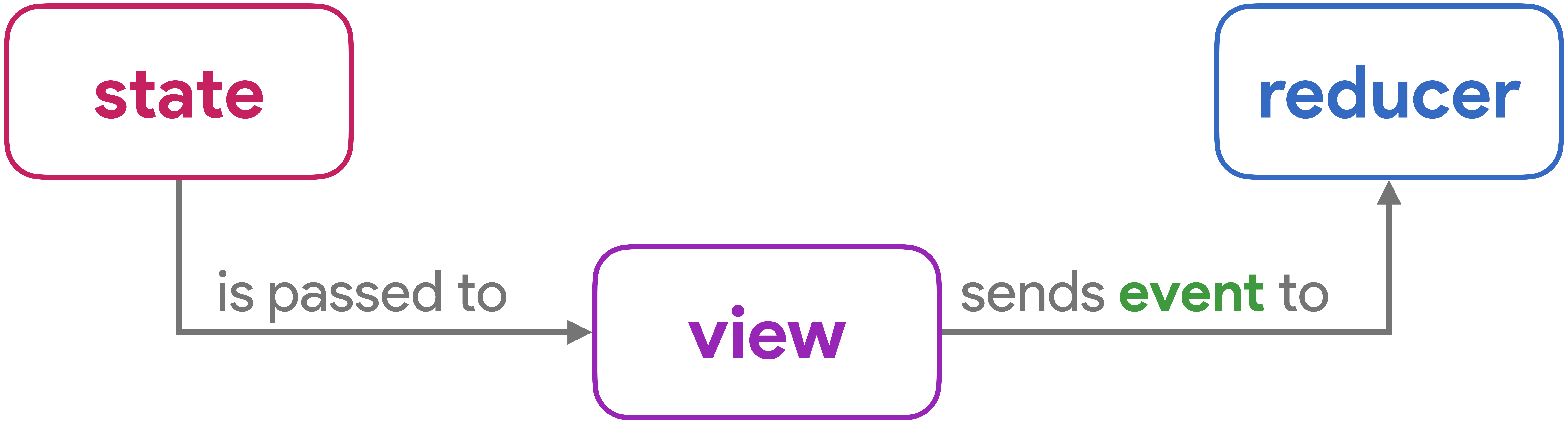
Unidirectional Data Flow

Unidirectional Data Flow

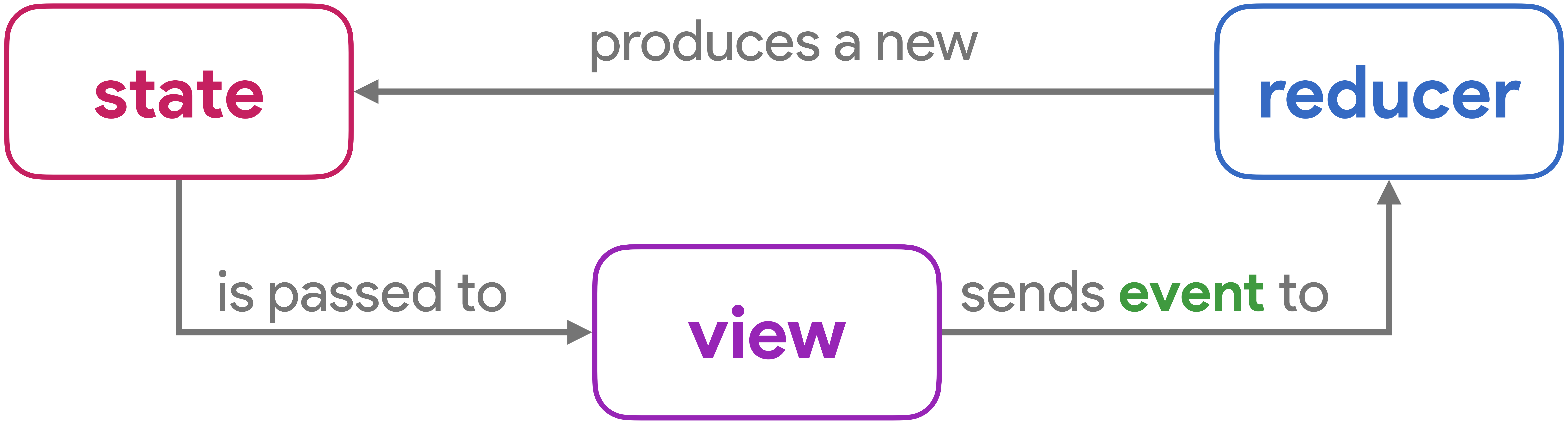
Unidirectional Data Flow



Unidirectional Data Flow



Unidirectional Data Flow



Unidirectional Data Flow

Unidirectional Data Flow

Redux

Unidirectional Data Flow

Redux

Bloc

Unidirectional Data Flow

Redux

Bloc

Cubit

Bloc

Bloc

Business logic component

Bloc

Business logic component

- A class containing a **state** and **reducers**.

Bloc

Business logic component

- A class containing a **state** and **reducers**.
- Uses the `Streams` API.

Bloc

Business logic component

- A class containing a **state** and **reducers**.
- Uses the `Streams` API.
- Manages and produces new **states** internally.

Bloc

Business logic component

- A class containing a **state** and **reducers**.
- Uses the `Streams` API.
- Manages and produces new **states** internally.
- Consumes **events** in the form of classes.

Bloc

Business logic component

- A class containing a **state** and **reducers**.
- Uses the `Streams` API.
- Manages and produces new **states** internally.
- Consumes **events** in the form of classes.

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

Bloc

Business logic component

- A class containing a **state** and **reducers**.
- Uses the `Streams` API.
- Manages and produces new **states** internally.
- Consumes **events** in the form of classes.

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

class CounterBloc extends
  Bloc<CounterEvent, CounterState>
```

Bloc

Bloc

- Are usually created per feature or screen.

Bloc

- Are usually created per feature or screen.
- Can be passed using `BlocProviders` with `package:flutter_bloc`.

Bloc

Bloc

Example

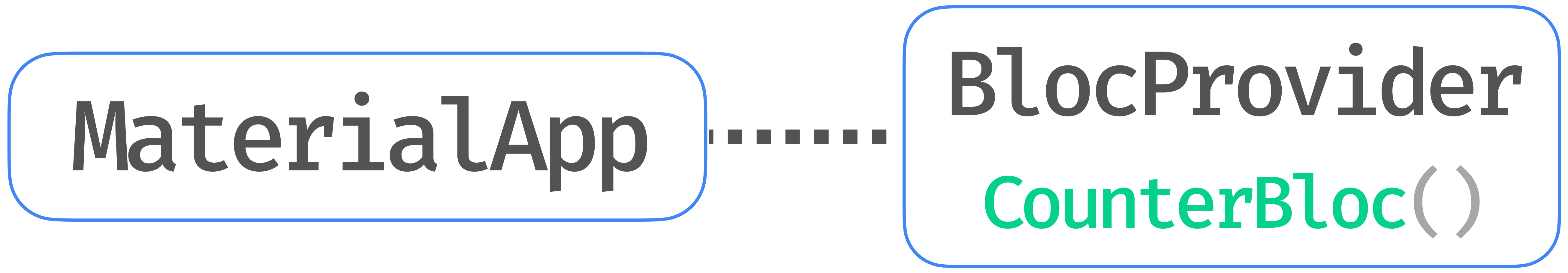
Bloc

Example

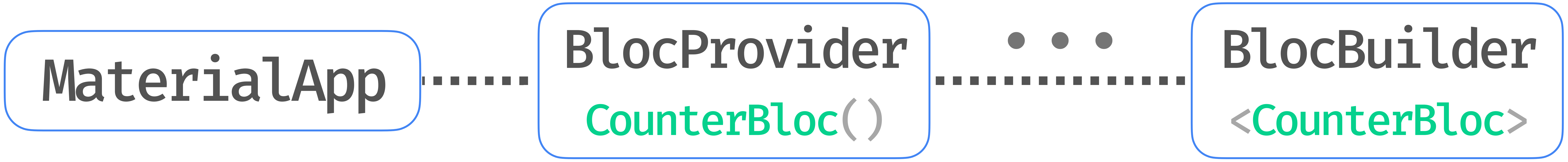
```
MaterialApp
```

Bloc

Example



Bloc
Example



Bloc

Example

MaterialApp
BlocProvider
CounterBloc()
BlocBuilder
<CounterBloc>

Cubit

Cubit

Bloc’s little brother

Cubit

Bloc’s little brother

- A smaller, simpler form of Bloc.

Cubit

Bloc’s little brother

- A smaller, simpler form of Bloc.
- All Blocs are actually cubits.

Cubit

Bloc’s little brother

- A smaller, simpler form of Bloc.
- All Blocs are actually cubits.
- Same concepts, different execution.

Cubit

Bloc’s little brother

- A smaller, simpler form of Bloc.
- All Blocs are actually cubits.
- Same concepts, different execution.
- Uses public methods instead of classes to convey **events**.

Cubit

Cubit in action

Google Developers

Experts

Thank You!

Slides and code available at https://cutt.ly/jfktalks

Jay (Jeroen Meijer)
Flutter & Dart GDE
@jfkdev