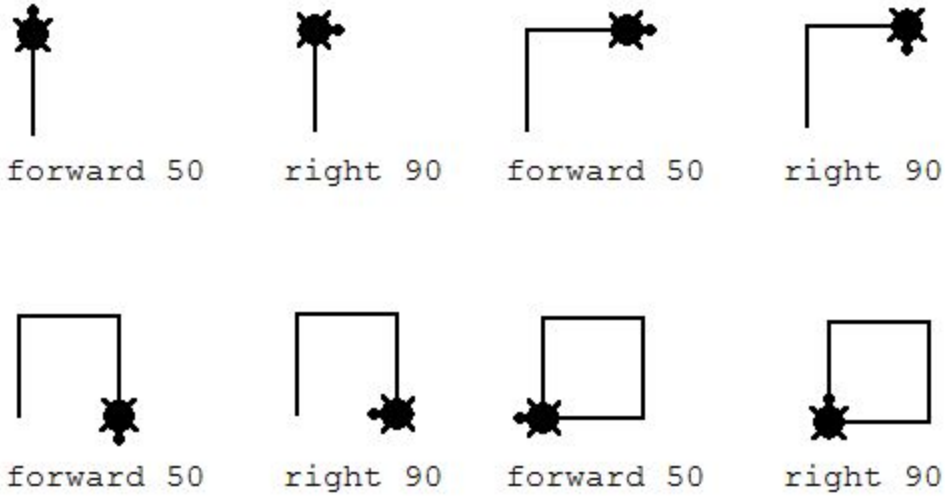
MS Excel

35 years of mobile development platform
with built in database functionality

Flutter State Management

(which is almost as good as MS Excel)



© 2000 Logo Foundation

```
CS
pu
rt 90
bk 150
lt 90
bk 150
rt 90
pd
```

```
repeat 4
[
 fd 300
 lt 90
]

lt 90
fd 300
rt 60
fd 174
```

```
rt 60
fd 174
lt 30

rt 90
fd 300
rt 90
fd 200
rt 90
fd 90
```

```
pu
fd 20
pd
fd 90
rt 90
fd 100
rt 90
fd 200
```

```
rt 90
fd 190
rt 90
pu
fd 75
rt 90
pd
```

```
repeat 4
[
 fd 75
 lt 90
]
fd 37.5
lt 90
fd 75
lt 90
fd 37.5
lt 90
fd 37.5
lt 90
fd 75
```

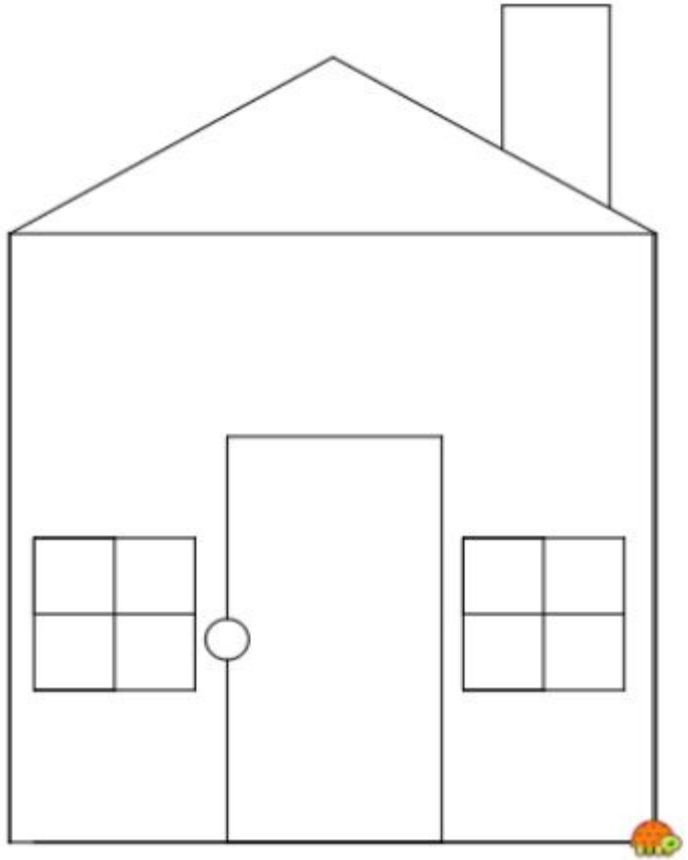
```
pu
rt 90
fd 37.5
fd 75
lt 90
fd 215
rt 180
fd 90
rt 90
pu
fd 75
rt 90
pd
```

```
repeat 4
[
 fd 75
 lt 90
]
fd 37.5
lt 90
fd 75
lt 90
fd 37.5
lt 90
fd 37.5
lt 90
fd 75
```

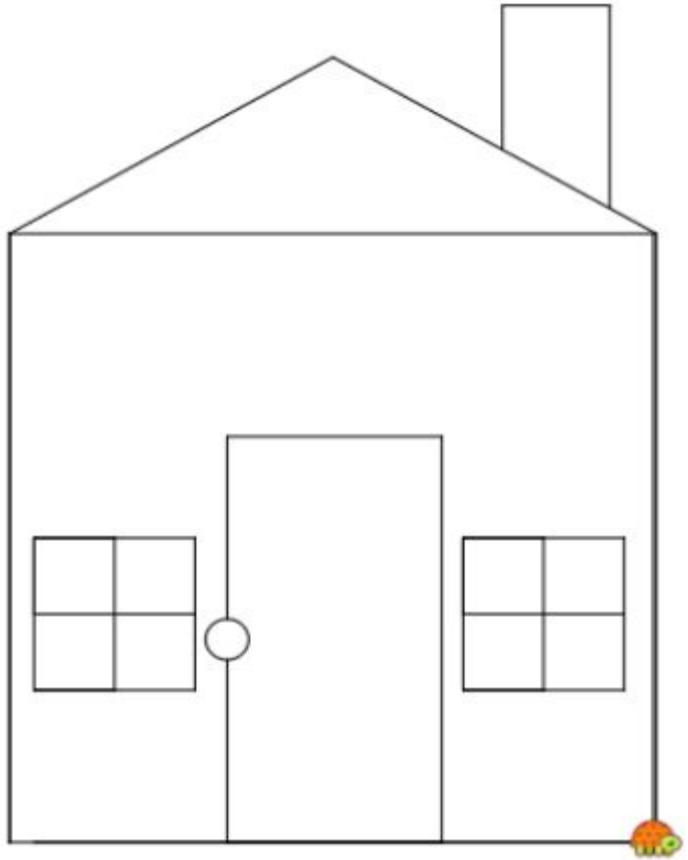
```
pu
rt 90
fd 37.5
fd 75
rt 90
fd 185
rt 90
fd 100
arc 360 10
rt 180
fd 100
```

```
rt 90
fd 100
rt 90
fd 300
rt 90
fd 300
lt 150
fd 25
rt 60
pd
fd 100
lt 90
```

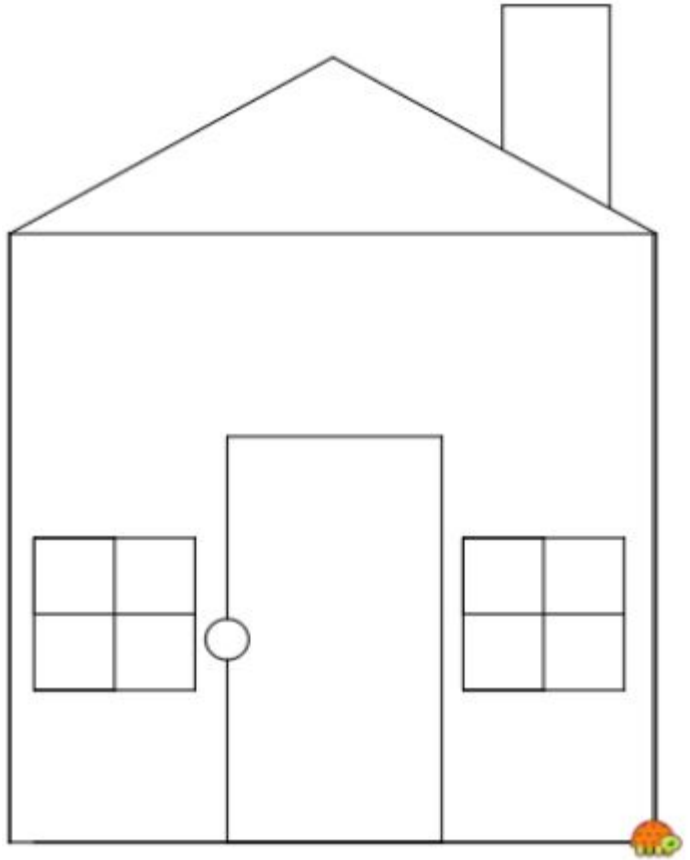
```
fd 50
lt 90
fd 70
pu
fd 340
lt 90
fd 70
```



```
<scene>
    <house>
        <roof>
            <chimney />
        </roof>
        <wall>
            <window />
            <door />
            <window />
        </wall>
    </house>
</scene>
```

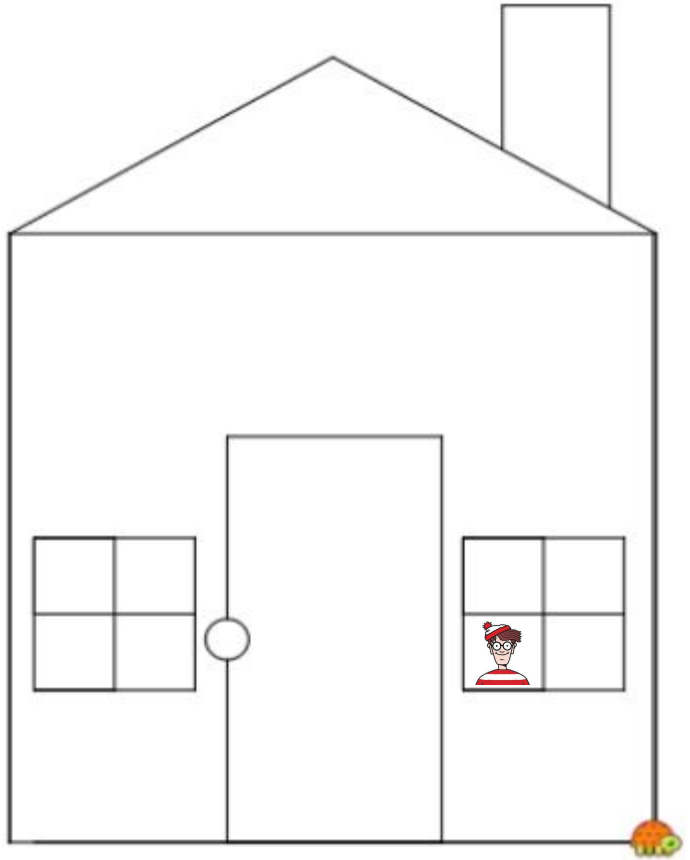


```
Scene(
    child: House(
        roof: Roof(chimney: true),
        wall: Wall(
            children: [
                Window(),
                Door(),
                Window()
            ]
        )
    )
);
```

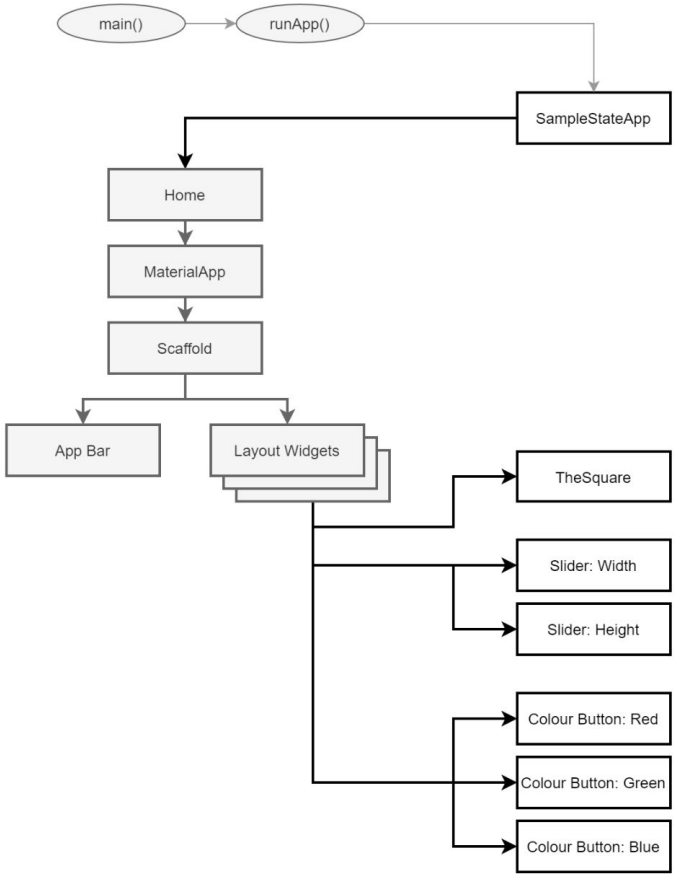


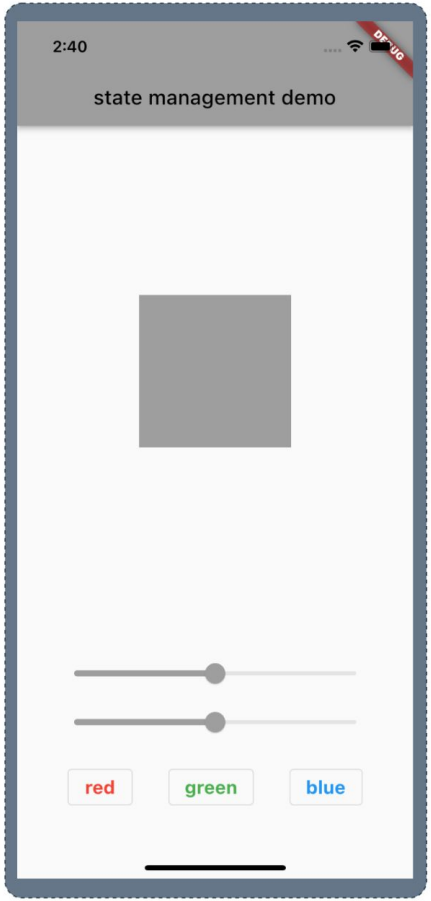
Where is the State?

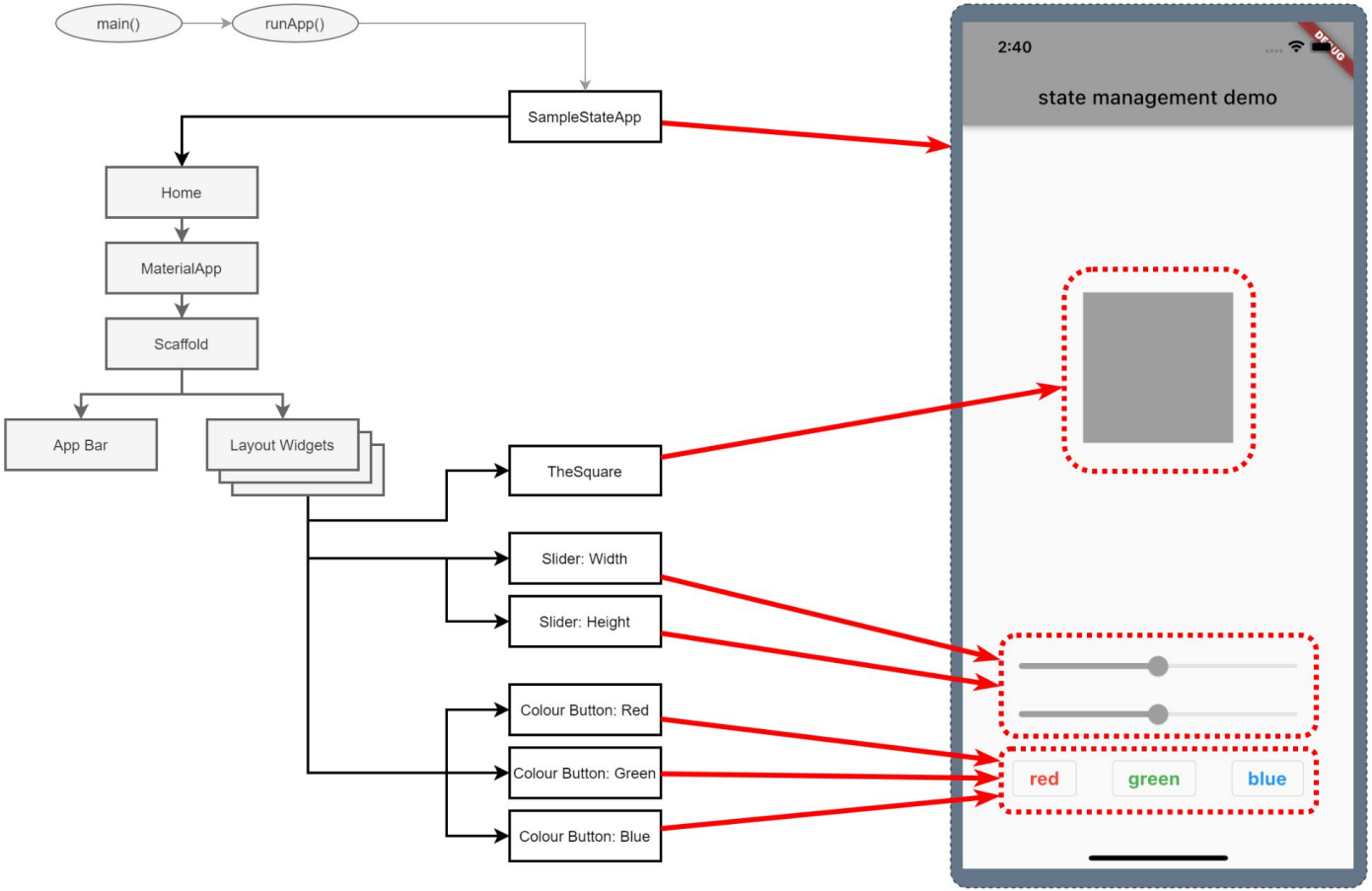
```
Scene(
    child: House(
        roof: Roof(chimney: true),
        wall: Wall(
            children: [
                Window(),
                Door(),
                Window()
            ]
        )
    )
);
```



What is state?


main()
runApp()
SampleStateApp
Home
MaterialApp
Scaffold
App Bar
Layout Widgets
TheSquare
Slider: Width
Slider: Height
Colour Button: Red
Colour Button: Green
Colour Button: Blue

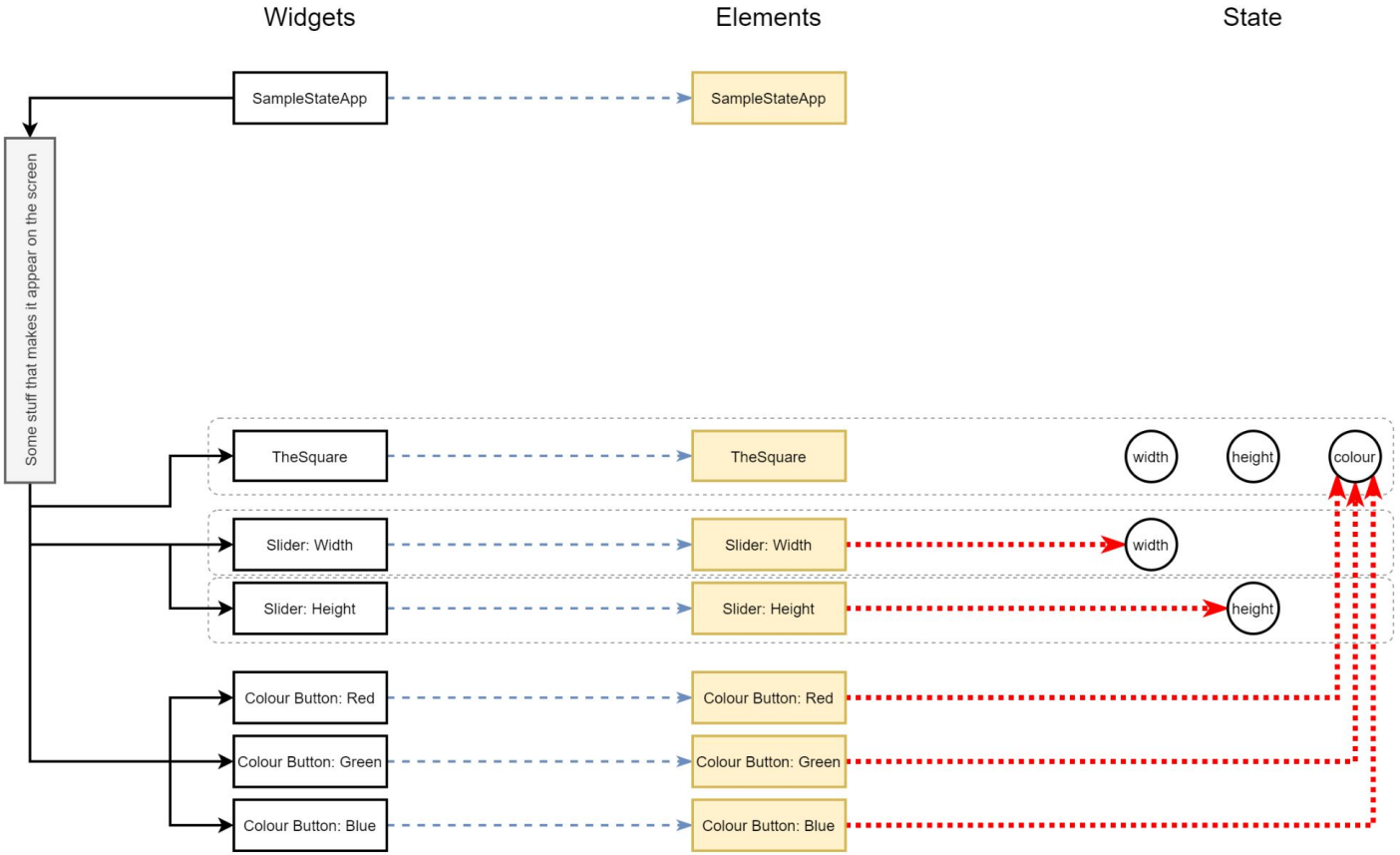

2:40
DEBUG
state management demo
red
green
blue


main()
runApp()
SampleStateApp
Home
MaterialApp
Scaffold
App Bar
Layout Widgets
TheSquare
Slider: Width
Slider: Height
Colour Button: Red
Colour Button: Green
Colour Button: Blue
2:40
state management demo
red
green
blue

“working” example

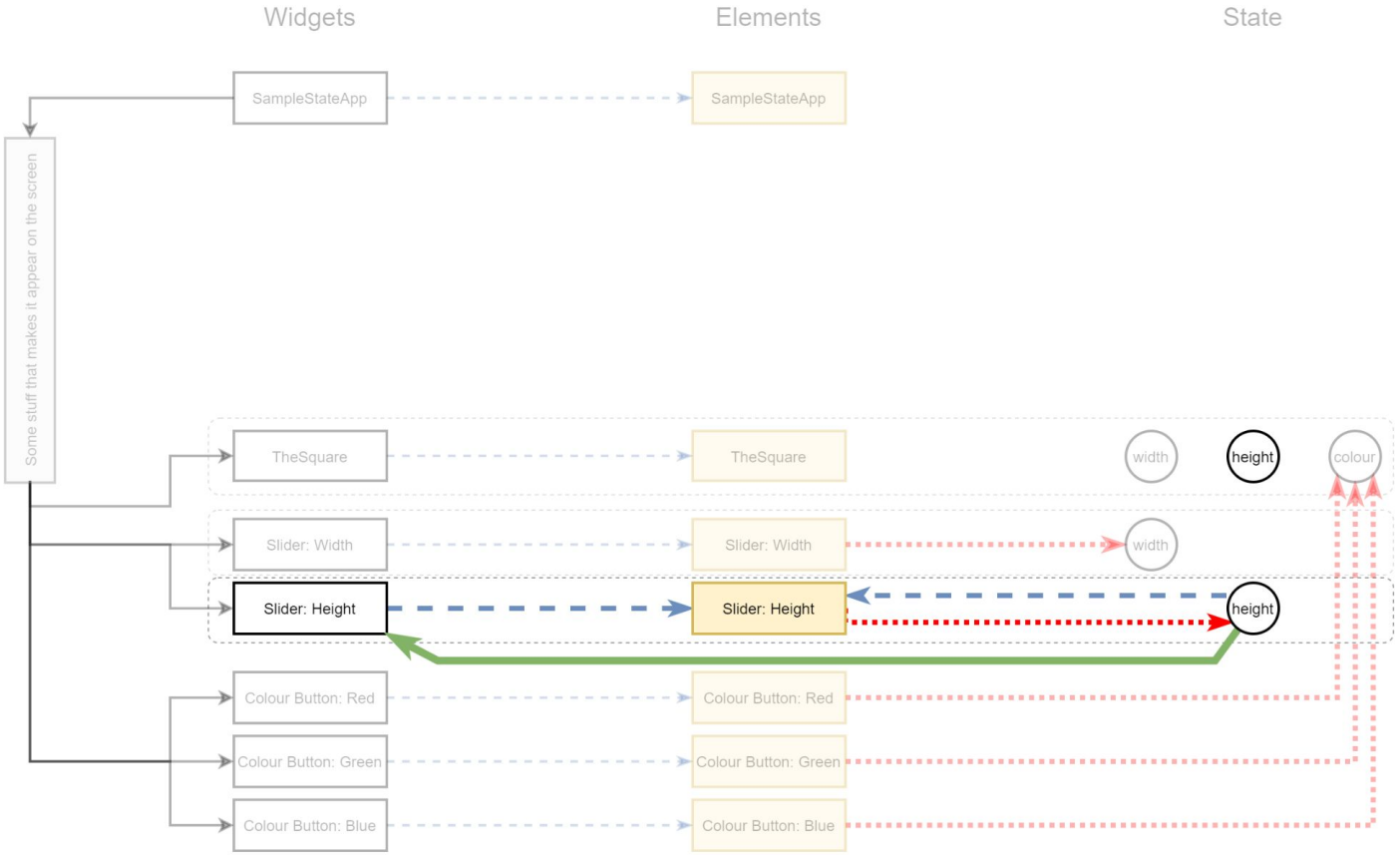

SampleStateApp
Some stuff that makes it appear on the screen
TheSquare
width
height
colour
Slider: Width
width
Slider: Height
height
Colour Button: Red
Colour Button: Green
Colour Button: Blue


SampleStateApp
Some stuff that makes it appear on the screen
TheSquare
width
height
colour
Slider: Width
width
Slider: Height
height
Colour Button: Red
Colour Button: Green
Colour Button: Blue


Widgets
Elements
State
SampleStateApp
SampleStateApp
Some stuff that makes it appear on the screen
TheSquare
TheSquare
width
height
colour
Slider: Width
Slider: Width
width
Slider: Height
Slider: Height
height
Colour Button: Red
Colour Button: Red
Colour Button: Green
Colour Button: Green
Colour Button: Blue
Colour Button: Blue

code

stateless to stateful

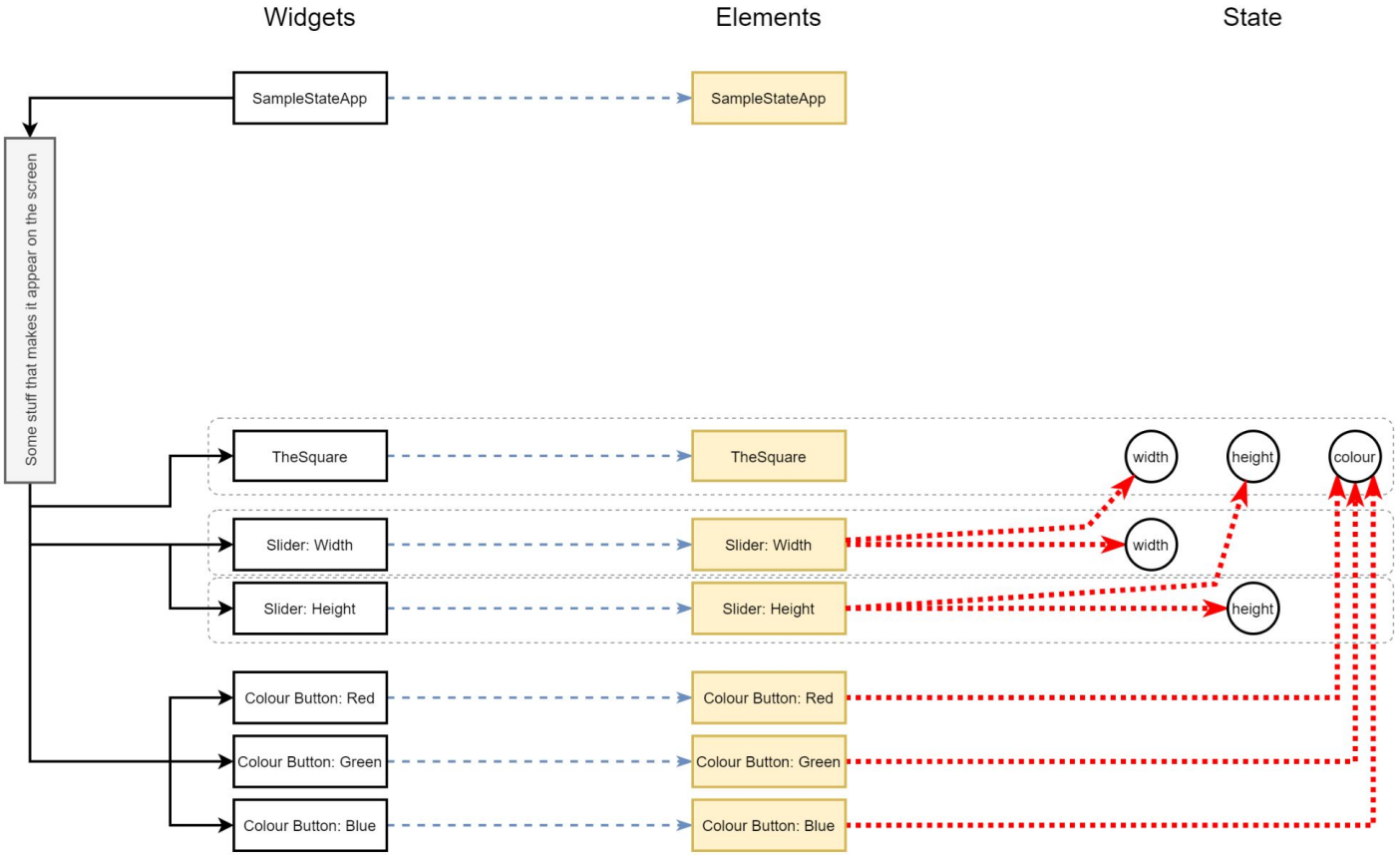

Widgets
Elements
State
SampleStateApp
SampleStateApp
Some stuff that makes it appear on the screen
TheSquare
TheSquare
width
height
colour
Slider: Width
Slider: Width
width
Slider: Height
Slider: Height
height
Colour Button: Red
Colour Button: Red
Colour Button: Green
Colour Button: Green
Colour Button: Blue
Colour Button: Blue

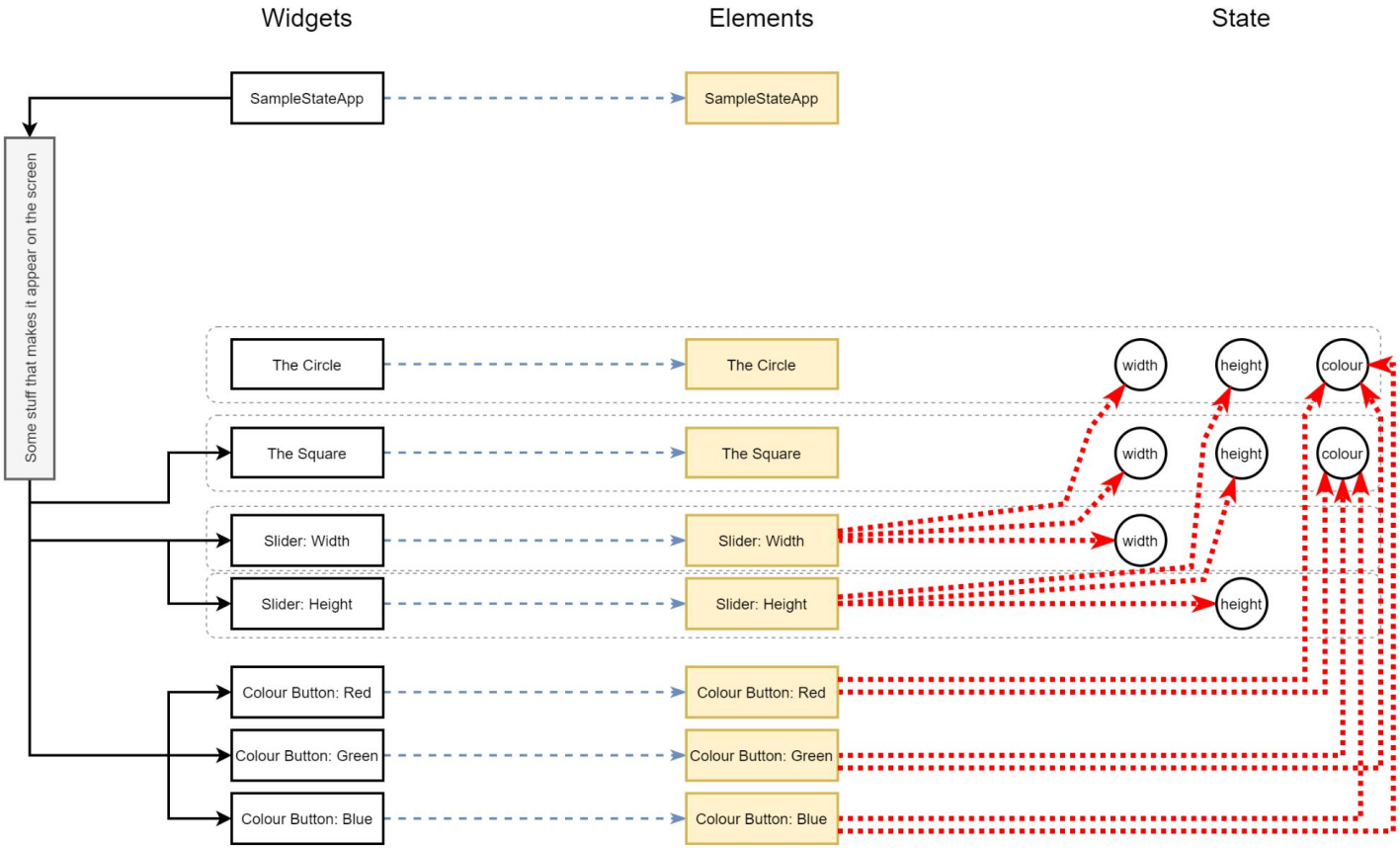
code

dirty nodes

Does State Need Management?

What is the problem?


Widgets
Elements
State
SampleStateApp
SampleStateApp
Some stuff that makes it appear on the screen
TheSquare
TheSquare
width
height
colour
Slider: Width
Slider: Width
width
Slider: Height
Slider: Height
height
Colour Button: Red
Colour Button: Red
Colour Button: Green
Colour Button: Green
Colour Button: Blue
Colour Button: Blue


Widgets
Elements
State
SampleStateApp
SampleStateApp
Some stuff that makes it appear on the screen
The Circle
The Circle
The Square
The Square
Slider: Width
Slider: Width
Slider: Height
Slider: Height
Colour Button: Red
Colour Button: Red
Colour Button: Green
Colour Button: Green
Colour Button: Blue
Colour Button: Blue
width
height
colour
width
height
colour
width
height

Contents

https://docs.flutter.dev/development/data-and-backend/state-mgmt/options

Contents

https://docs.flutter.dev/development/data-and-backend/state-mgmt/options

Contents

https://docs.flutter.dev/development/data-and-backend/state-mgmt/options

code

https://github.com/michalporeba/flutter-state-show

Side by side,
bit by bit
comparison

State

Scoped State

```
class TheState extends Model {
  double _width = 0.5;
  double _height = 0.5;
  Color _color = Colors.grey;

  double get width => _width;
  double get height => _height;
  Color get color => _color;

  set width(double value) {
    _width = value;
    notifyListeners();
  }

  set height(double value) {
    _height = value;
    notifyListeners();
  }

  set color(Color value) {
    _color = value;
    notifyListeners();
  }
}
```

Redux, BLoC and Real BLoC

```
class TheState {
  final double width;
  final double height;
  final Color color;

  const TheState({
    required this.width,
    required this.height,
    required this.color
  });

  const TheState.initial(): width = 0.5, height= 0.5, color = Colors.grey;

  TheState copyWith({
    double? width,
    double? height,
    Color? color,
  }) => TheState(
      width: width ?? this.width,
      height: height ?? this.height,
      color: color ?? this.color
  );

  double getSide(String attribute)
  => (attribute == 'width')
      ? width
      : height;
}
```

Scoped State

```
class TheState extends Model {
  double _width = 0.5;
  double _height = 0.5;
  Color _color = Colors.grey;

  double get width => _width;
  double get height => _height;
  Color get color => _color;

  set width(double value) {
    _width = value;
    notifyListeners();
  }

  set height(double value) {
    _height = value;
    notifyListeners();
  }

  set color(Color value) {
    _color = value;
    notifyListeners();
  }
}
```

Redux, BLoC and Real BLoC

```
class TheState {
  final double width;
  final double height;
  final Color color;

  const TheState({
    required this.width,
    required this.height,
    required this.color
  });

  const TheState.initial(): width = 0.5, height= 0.5, color = Colors.grey;

  TheState copyWith({
    double? width,
    double? height,
    Color? color,
  }) => TheState(
      width: width ?? this.width,
      height: height ?? this.height,
      color: color ?? this.color
  );

  double getSide(String attribute)
  => (attribute == 'width')
      ? width
      : height;
}
```

Scoped State

```
class TheState extends Model {
  double _width = 0.5;
  double _height = 0.5;
  Color _color = Colors.grey;

  double get width => _width;
  double get height => _height;
  Color get color => _color;

  set width(double value) {
    _width = value;
    notifyListeners();
  }

  set height(double value) {
    _height = value;
    notifyListeners();
  }

  set color(Color value) {
    _color = value;
    notifyListeners();
  }
}
```

Redux, BLoC and Real BLoC

```
class TheState {
```

```
const TheState({
  required this.width,
  required this.height,
  required this.color
});
```

ey;

freezed 1.1.1

Published 41 days ago• dash-overflow.net Null safety

SDK | DART

1.43K

Readme | Changelog | Example | Installing | Versions | Scores

Build passing | pub v1.1.1 | chat 118 online

}

Welcome to Freezed, yet another code generator for unions/pattern-matching/copy.

State Manipulation

Redux

```
abstract class StateAction {
  const StateAction();
  TheState modify(TheState state);
}

class SetSide extends StateAction {
  final String attribute;
  final double size;
  const SetSide(this.attribute, this.size);

  @override
  TheState modify(TheState state) {
    return attribute == 'width'
        ? state.copyWith(width: size)
        : state.copyWith(height: size);
  }
}

class SetColor extends StateAction {
  final Color color;
  const SetColor(this.color);
  @override
  TheState modify(TheState state) => state.copyWith(color: color);
}

TheState myReducer(TheState state, dynamic action)
  => action.modify(state);

final store = Store<TheState>(
    myReducer,
    initialState: const TheState.initial()
);
```

Real BLoC

```
abstract class DemoEvent {
  const DemoEvent();
  TheState modify(TheState state);
}

class SetSide extends DemoEvent {
  final String attribute;
  final double size;
  const SetSide(this.attribute, this.size);

  @override
  TheState modify(TheState state) {
    return attribute == 'width'
      ? state.copyWith(width: size)
        : state.copyWith(height: size);
  }
}

class SetColor extends DemoEvent {
  final Color color;
  const SetColor(this.color);

  @override modify(TheState state) {
    return state.copyWith(color: color);
  }
}

class StateBloc extends Bloc<DemoEvent, TheState> {
  StateBloc(): super(const TheState.initial()) {
    on<DemoEvent>((event, emit) => emit(event.modify(state)));
  }
}
```

Redux

```
abstract class StateAction {
  const StateAction();
  TheState modify(TheState state);
}

class SetSide extends StateAction {
  final String attribute;
  final double size;
  const SetSide(this.attribute, this.size);

  @override
  TheState modify(TheState state) {
    return attribute == 'width'
        ? state.copyWith(width: size)
        : state.copyWith(height: size);
  }
}

class SetColor extends StateAction {
  final Color color;
  const SetColor(this.color);
  @override
  TheState modify(TheState state) => state.copyWith(color: color);
}

TheState myReducer(TheState state, dynamic action)
  => action.modify(state);

final store = Store<TheState>(
    myReducer,
    initialState: const TheState.initial()
);
```

Real BLoC

```
abstract class DemoEvent {
  const DemoEvent();
  TheState modify(TheState state);
}

class SetSide extends DemoEvent {
  final String attribute;
  final double size;
  const SetSide(this.attribute, this.size);

  @override
  TheState modify(TheState state) {
    return attribute == 'width'
      ? state.copyWith(width: size)
        : state.copyWith(height: size);
  }
}

class SetColor extends DemoEvent {
  final Color color;
  const SetColor(this.color);

  @override modify(TheState state) {
    return state.copyWith(color: color);
  }
}

class StateBloc extends Bloc<DemoEvent, TheState> {
  StateBloc(): super(const TheState.initial()) {
    on<DemoEvent>((event, emit) => emit(event.modify(state)));
  }
}
```

Redux


```
TheState myReducer(TheState state, dynamic action)
  => action.modify(state);

final store = Store<TheState>(
    myReducer,
    initialState: const TheState.initial()
);
```

Real BLoC


```
class StateBloc extends Bloc<DemoEvent, TheState> {
  StateBloc(): super(const TheState.initial()) {
    on<DemoEvent>((event, emit) => emit(event.modify(state)));
  }
}
```

Redux

```
abstract class StateAction {
  const StateAction();
  TheState modify(TheState state);
}

class SetSide extends StateAction {
  final String attribute;
  final double size;
  const SetSide(this.attribute, this.size);

  @override
  TheState modify(TheState state) {
    return attribute == 'width'
        ? state.copyWith(width: size)
        : state.copyWith(height: size);
  }
}

class SetColor extends StateAction {
  final Color color;
  const SetColor(this.color);
  @override
  TheState modify(TheState state) => state.copyWith(color: color);
}

TheState myReducer(TheState state, dynamic action)
  => action.modify(state);

final store = Store<TheState>(
    myReducer,
    initialState: const TheState.initial()
);
```

BLoC

```
class StateCubit extends Cubit<TheState> {
  StateCubit(): super(const TheState.initial());

  void setSide(String attribute, double size)
    => emit(
        (attribute == 'width')
            ? state.copyWith(width: size)
            : state.copyWith(height: size)
    );

  void setColor(Color color) => emit(state.copyWith(color: color));
}
```

Where is Scoped State Management?

The Slider

A **Stateless** Widget

Scoped State

```
return ScopedModelDescendant<TheState>(
    builder: (context, child, state) {
        return Slider(
            value: attribute == 'width' ? state.width : state.height,
            onChanged: (value) {
                if (attribute == 'width') {
                    state.width = value;
                } else {
                    state.height = value;
                }
            }
        );
    }
);
```

Redux

```
return StoreBuilder<TheState>(
    builder: (context, store) {
        return Slider(
            value: store.state.getSide(attribute),
            onChanged: (value) {
                store.dispatch(SetSide(attribute, value));
            }
        );
    }
);
```

BLoC

```
return BlocBuilder<StateCubit, TheState>(
    builder: (context, state) {
        return Slider(
            value: state.getSide(attribute),
            onChanged: (value) {
                context.read<StateCubit>()
                    .setSide(attribute, value);
            }
        );
    }
);
```

Real BLoC

```
return BlocBuilder<StateBloc, TheState>(
    builder: (context, state) {
        return Slider(
            value: state.getSide(attribute),
            onChanged: (value) {
                context.read<StateBloc>()
                    .add(SetSide(attribute, value));
            }
        );
    }
);
```

The Hoisted Scope

The Starting Point Comparison

What is inside the

```
void main() {
  runApp(...);
}
```

The Starting Point Comparison

```
ScopedModel<TheState>(
    model: TheState(),
    child: Home()
);
```

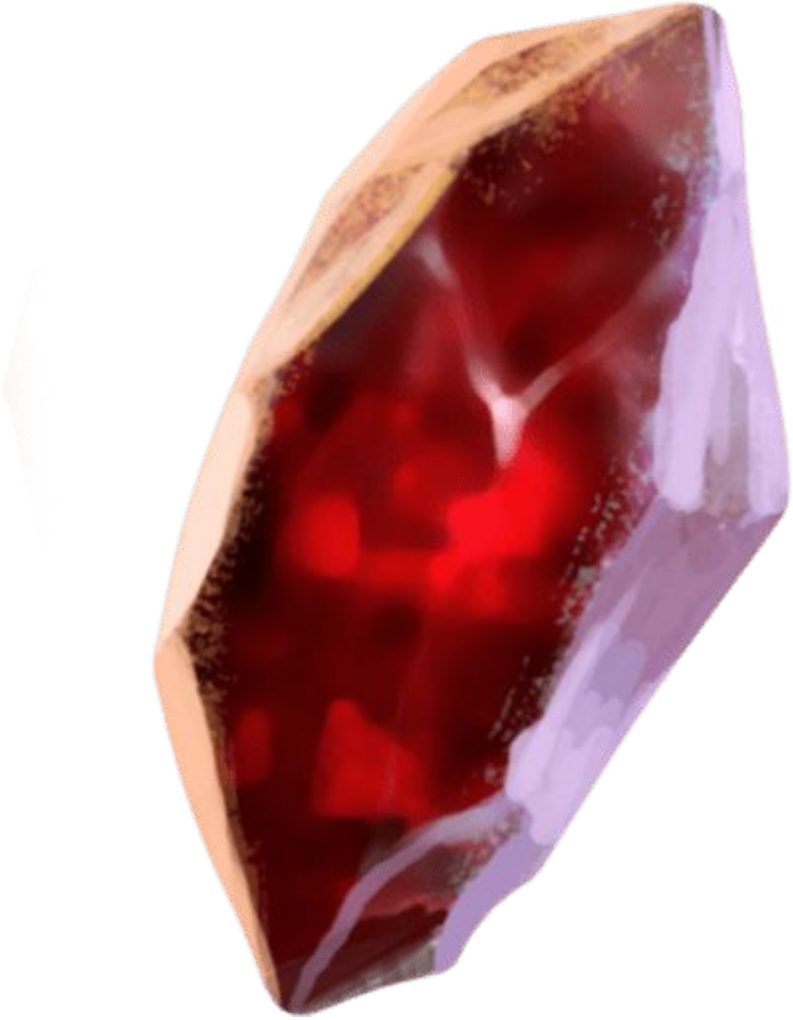
```
StoreProvider<TheState>(
    store: store,
    child: Home()
);
```

```
BlocProvider(
    create: (_) => StateCubit(),
    child: Home()
);
```

```
BlocProvider(
    create: (_) => StateBloc(),
    child: home
);
```

Are they all the same?

Or, what is the difference and why should we care?



Scoped State

GetIt

GetX

Provider

Riverpod

- **Hoist the scope to the top**
- **Get it when needed**

Redux

Fish-Redux

MobX

- **Central state**
- **Unidirectional flow (Flux)**
- **Time-travel**

2015, Dan Abramov and Andrew Clark

BLoC

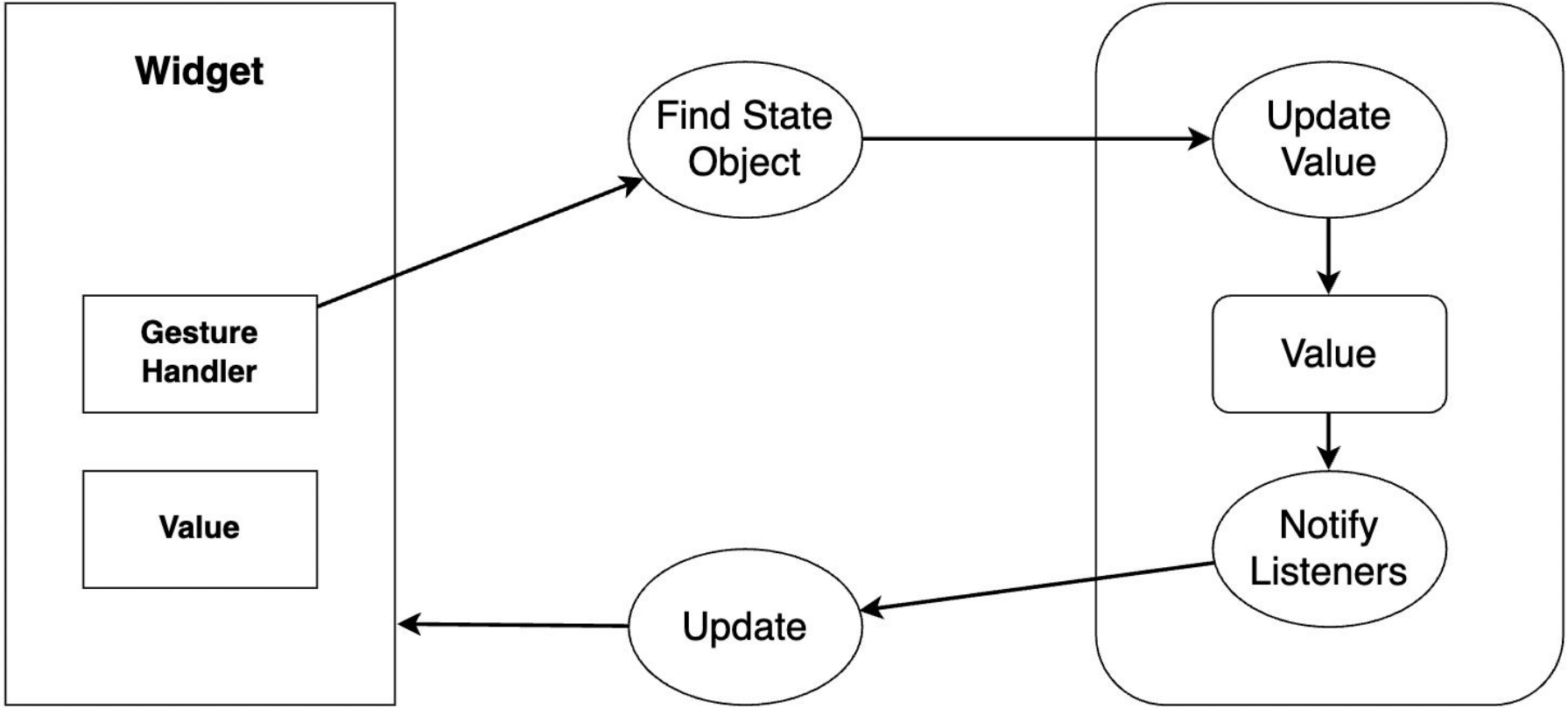
Flutter Commands

Triple Pattern (Segmented State Pattern)

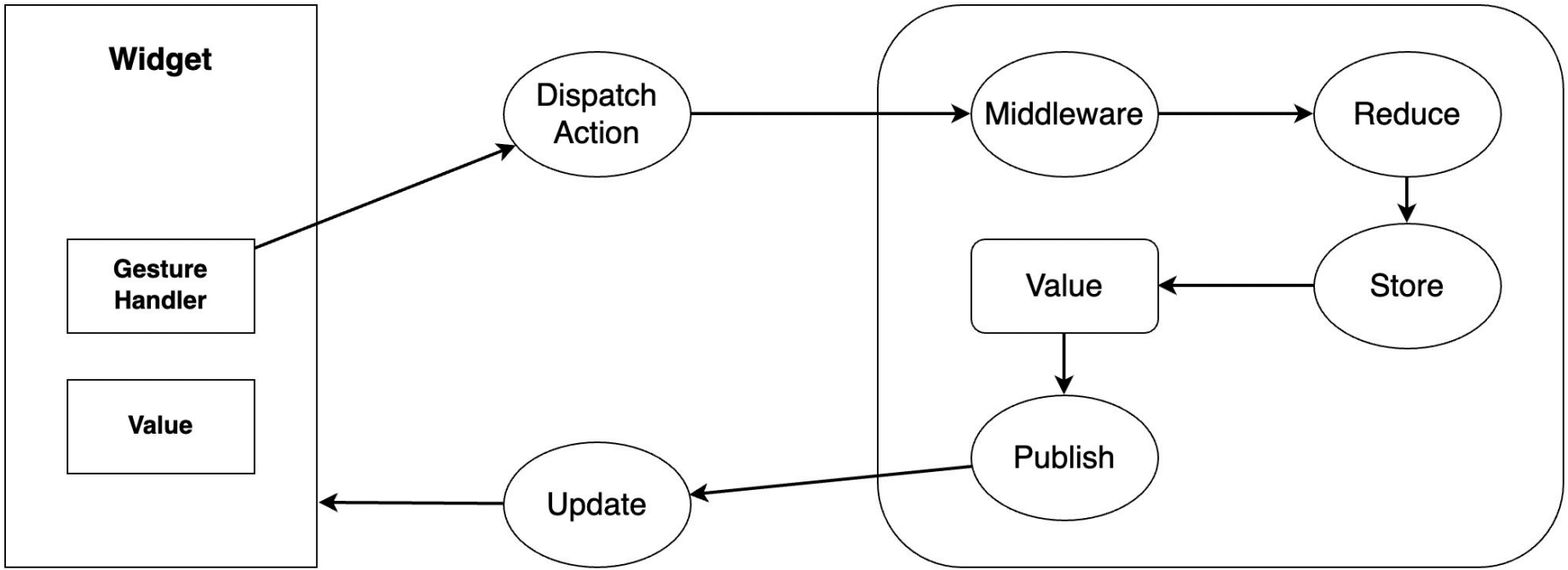
- **Streams**
- **Flow is not unidirectional**

2018, Google, a Dart thing

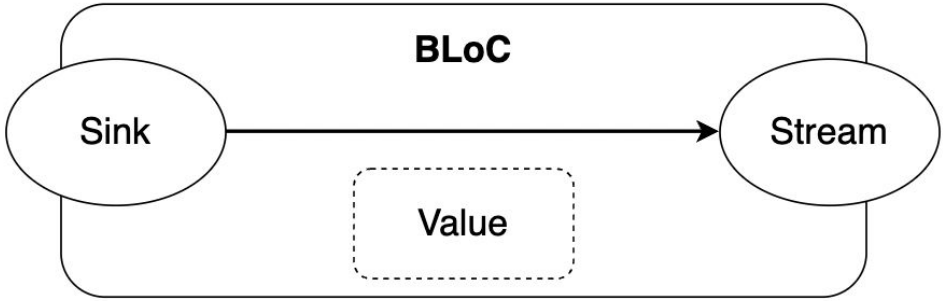
Scoped State

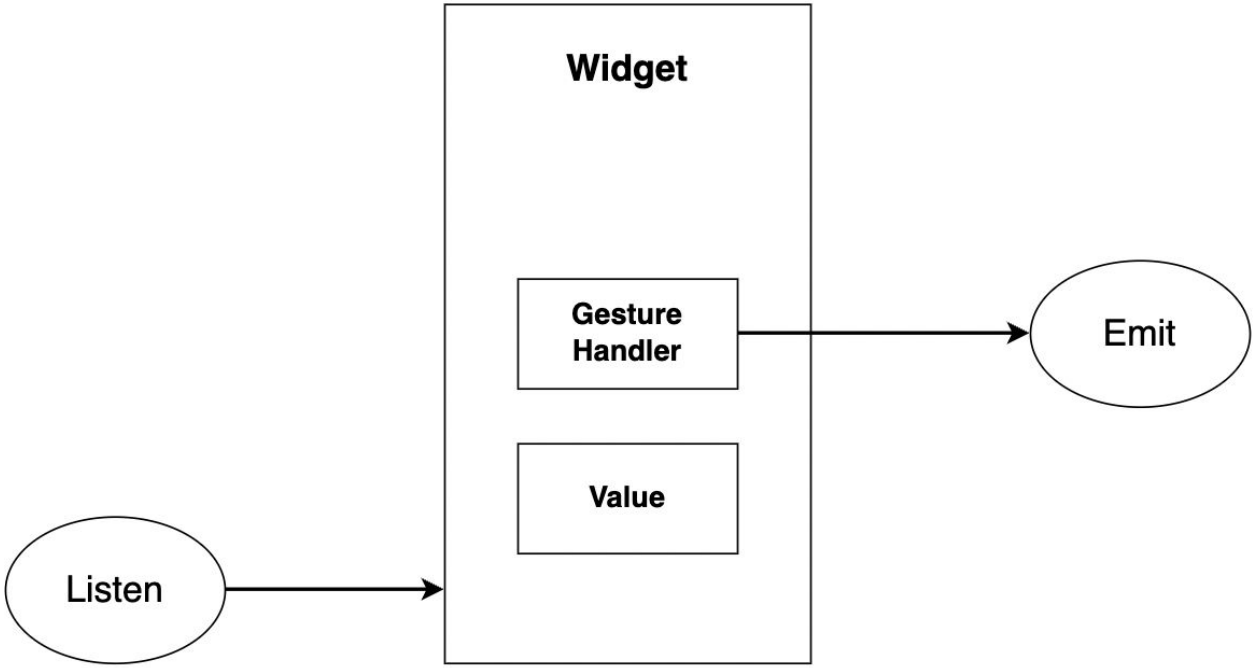


Redux


Widget
Gesture Handler
Value
Dispatch Action
Update
Middleware
Reduce
Store
Value
Publish

BLoC


BLoC
Sink
Stream
Value


Widget
Gesture Handler
Emit
Value
Listen

Scoped State

- Simple to understand
- Simple to implement

- Depends on mutable state
- Helps with access, but doesn't do much about state transition.

Redux

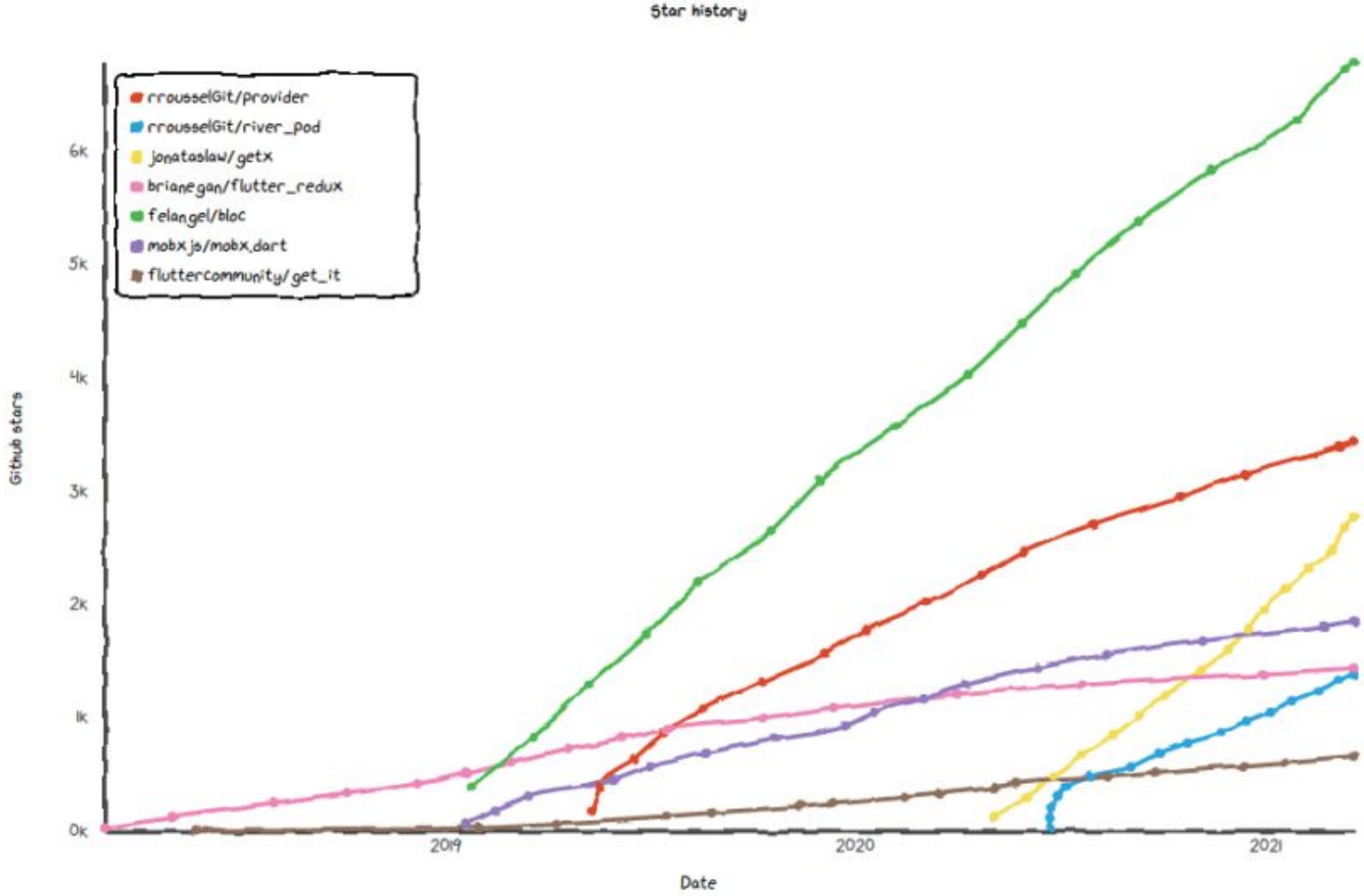
- Makes sense
- Easy to debug
- Time-travel
- Middleware
- Good for distributed event systems

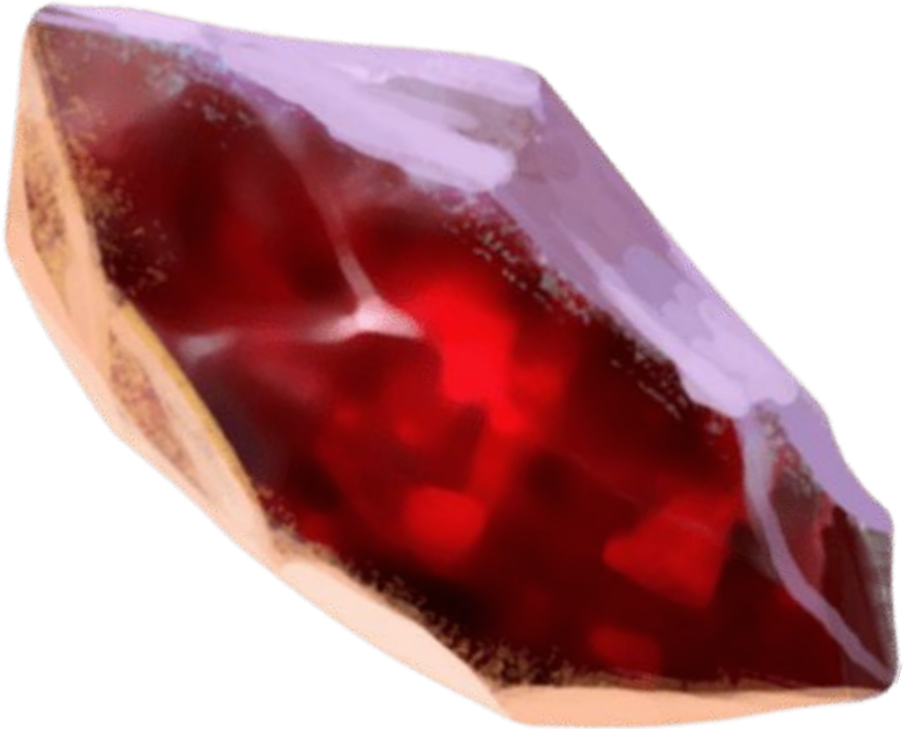
- Single, immutable state
  - encapsulation
  - memory

BLoC

- Divide and conquer state
- More aligned with user interaction flow
- Easy to test

- Can be difficult to start
- Can be difficult later too
- It's just a style
  - many decisions
  - things can go wrong


Star history
rrousselGit/provider
rrousselGit/river_pod
jonataslaw/getx
brianegan/flutter_redux
felangel/bloc
mobx.js/mobx.dart
fluttercommunity/get_it
6k
5k
4k
3k
2k
1k
0k
Github stars
2019
2020
2021
Date



Have I missed anything?

What is the **StatefulWidget** for?

MS Excel

could do it all too
(with enough VBA scripting)
but the choice is yours to make