Flutter
State management - How to Redux?

Paulina Szklarska
Google Developer Expert in Flutter
Flutter Developer (ex-Android Developer)

@p_szklarska
@pszklarska
@pszklarska

Flutter!

Who’s in? 🙋‍♀️ 🙋‍♂️

It’s all about **Widgets**

imperative style **UI**

~~imperative style **UI**~~

declarative style **UI**

```kotlin
class MainActivity : AppCompatActivity {

    override fun onCreate(
            savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)
    }

}
```

```dart
class App extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<App> {

  @override
  Widget build(BuildContext context) {
    return Container(
      child: FlatButton(
      ),
    );
  }
}
```

```kotlin
class MainActivity : AppCompatActivity {

    override fun onCreate(
            savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        button.text = "Tap me!"
    }

}
```

```dart
class App extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<App> {

  @override
  Widget build(BuildContext context) {
    return Container(
      child: FlatButton(
        child: Text('Tap me!'),
      ),
    );
  }
}
```

```kotlin
class MainActivity : AppCompatActivity {

    override fun onCreate(
            savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        button.text = "Tap me!"
        button.setOnClickListener {

        }
    }

}
```

```dart
class App extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<App> {

  @override
  Widget build(BuildContext context) {
    return Container(
      child: FlatButton(
        child: Text('Tap me!'),
        onPressed: () {

        },
      ),
    );
  }
}
```

```kotlin
class MainActivity : AppCompatActivity {

    override fun onCreate(
            savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        button.text = "Tap me!"
        button.setOnClickListener {
            button.setBackgroundColor(
                    Color.BLUE)
        }
    }

}
```

```dart
class App extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<App> {

  @override
  Widget build(BuildContext context) {
    return Container(
      child: FlatButton(
        child: Text('Tap me!'),
        onPressed: () {

        },
      ),
    );
  }
}
```

```kotlin
class MainActivity : AppCompatActivity {

    override fun onCreate(
            savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        button.text = "Tap me!"
        button.setOnClickListener {
            button.setBackgroundColor(
                    Color.BLUE)
        }
    }

}
```

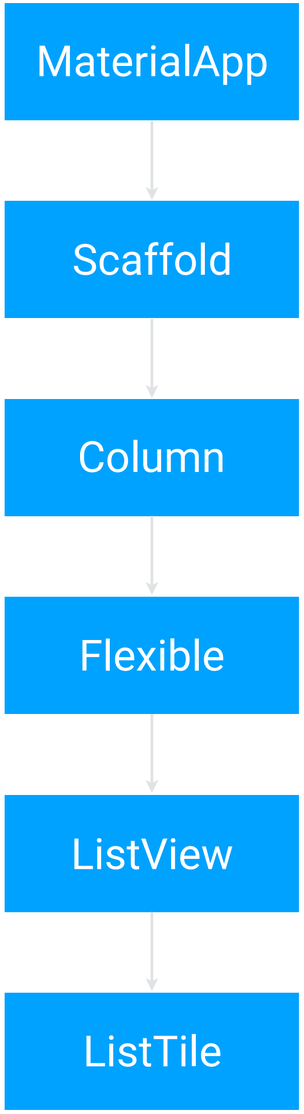
```dart
class App extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<App> {
  var buttonColor = Colors.white;

  @override
  Widget build(BuildContext context) {
    return Container(
      child: FlatButton(
        child: Text('Tap me!'),
        onPressed: () {

        },
        color: buttonColor,
      ),
    );
  }
}
```

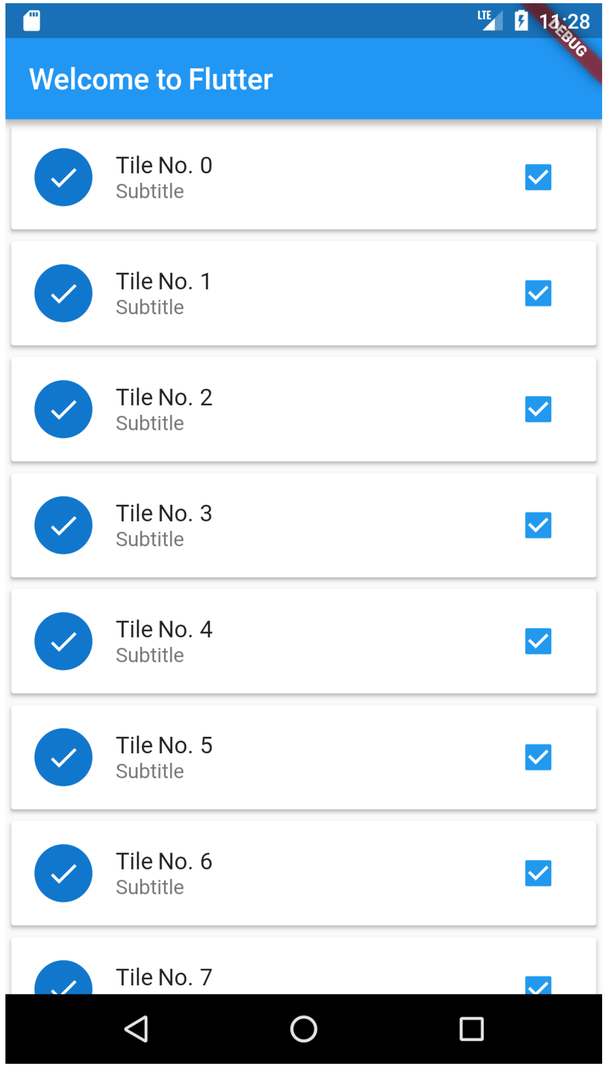
```kotlin
class MainActivity : AppCompatActivity {

    override fun onCreate(
            savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        button.text = "Tap me!"
        button.setOnClickListener {
            button.setBackgroundColor(
                    Color.BLUE)
        }
    }

}
```

```dart
class App extends StatefulWidget {
  @override
  _AppState createState() => _AppState();
}

class _AppState extends State<App> {
  var buttonColor = Colors.white;

  @override
  Widget build(BuildContext context) {
    return Container(
      child: FlatButton(
        child: Text('Tap me!'),
        onPressed: () {
          setState(() {
            buttonColor = Colors.blue;
          });
        },
        color: buttonColor,
      ),
    );
  }
}
```

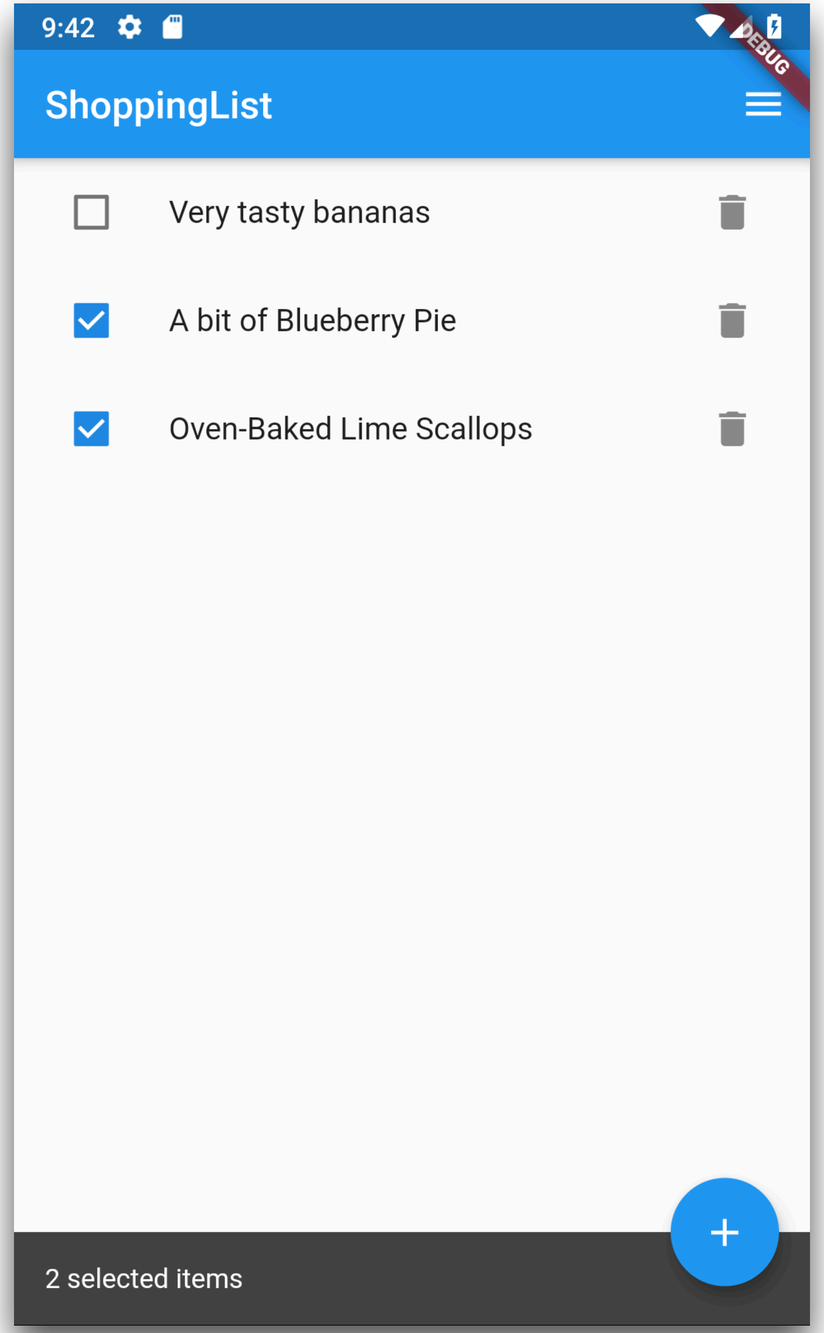
It’s all about **State**

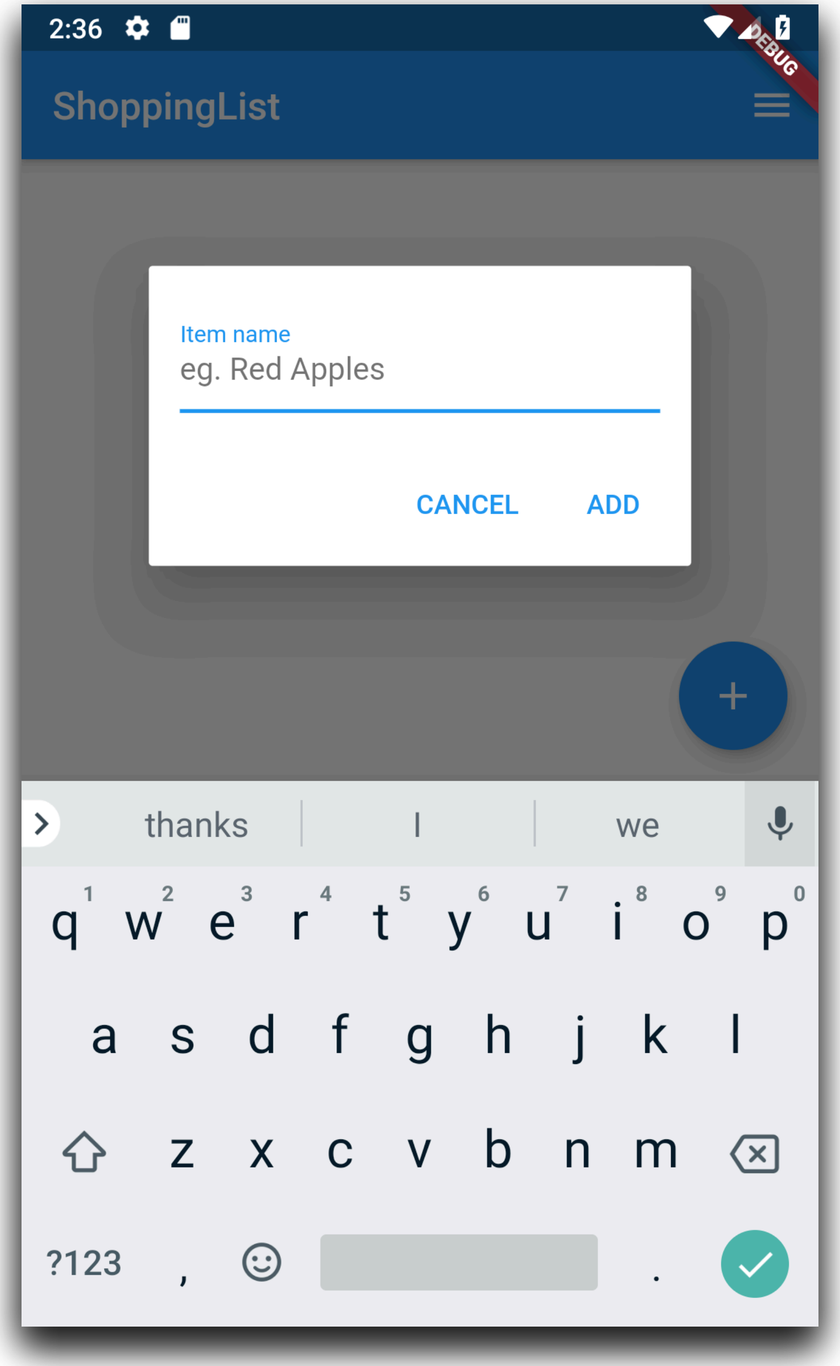
It’s all about **Widgets**


MaterialApp
Scaffold
Column
Flexible
ListView
ListTile


11:28
DEBUG
Welcome to Flutter
Tile No. 0
Subtitle
Tile No. 1
Subtitle
Tile No. 2
Subtitle
Tile No. 3
Subtitle
Tile No. 4
Subtitle
Tile No. 5
Subtitle
Tile No. 6
Subtitle
Tile No. 7

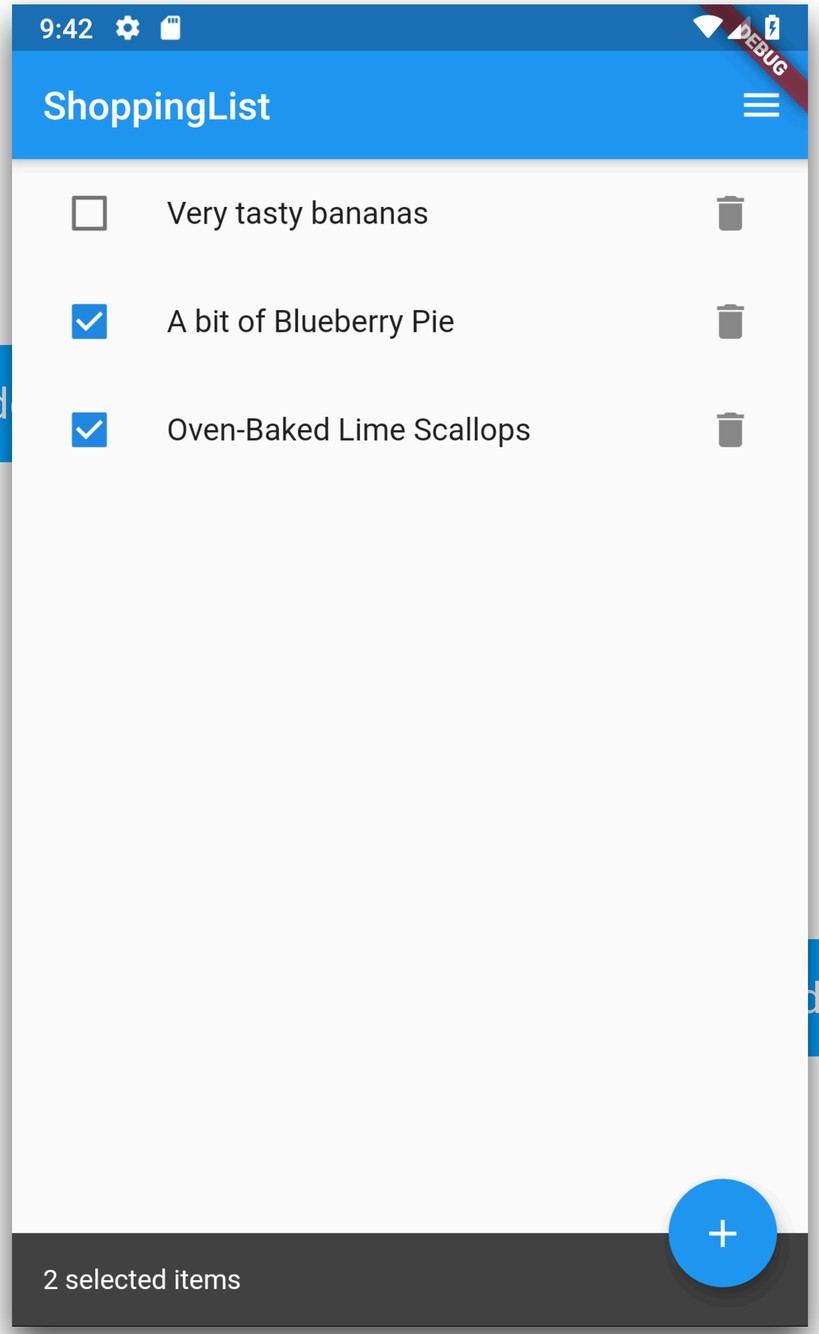
What is **the problem**?


9:42
DEBUG
ShoppingList
Very tasty bananas
A bit of Blueberry Pie
Oven-Baked Lime Scallops
2 selected items

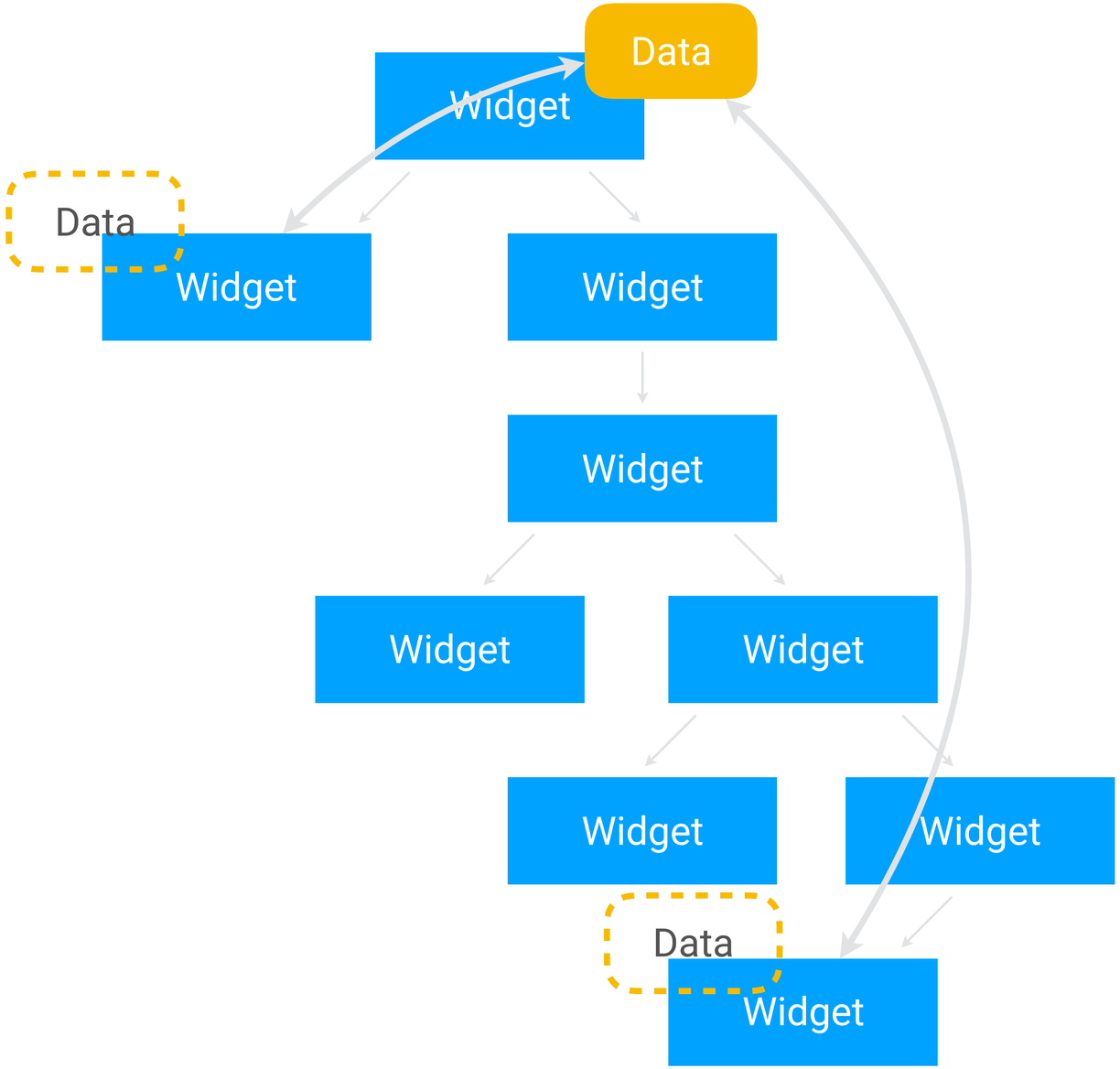

2:36
DEBUG
ShoppingList
Item name
eg. Red Apples
CANCEL
ADD
thanks
I
we
?123

Data

Wid



dget


Data
Widget
Data
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Data
Widget

```
class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ListView.builder(

    );
  }
}
```

```
class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: cartItems.length,

    );
  }
}
```

```
class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: cartItems.length,
      itemBuilder: (context, position) =>
          ShoppingListItem(cartItems[position]),
    );
  }
}
```

What are our options?

option A
pass data through
all the Widgets

```dart
class MyApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const MyApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingApp();
  }a
}a
```

```dart
class MyApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const MyApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingApp();
  }a
}a

class ShoppingApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return ShoppingList();
  }b
}b
```

```dart
class MyApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const MyApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingApp(cartItems);
  }a
}a

class ShoppingApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const ShoppingApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingList();
  }b
}b
```

```dart
class MyApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const MyApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingApp(cartItems);
  }
}

class ShoppingApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const ShoppingApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingList();
  }
}

class ShoppingList extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
   return ListView.builder(
      itemCount: cartItems.length,
      itemBuilder: (context, position) =>
          ShoppingListItem(cartItems[position]),
    );
  }
}
```

```dart
class MyApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const MyApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingApp(cartItems);
  }
}

class ShoppingApp extends StatelessWidget {
  final List<CartItem> cartItems;

  const ShoppingApp(this.cartItems);

  @override
  Widget build(BuildContext context) {
    return ShoppingList(cartItems);
  }
}

class ShoppingList extends StatelessWidget {
  final List<CartItem> cartItems;

  const ShoppingList(this.cartItems);

  @override
  Widget build(BuildContext context) {
   return ListView.builder(
      itemCount: cartItems.length,
      itemBuilder: (context, position) =>
          ShoppingListItem(cartItems[position]),
    );
  }
}
```

What are our options?

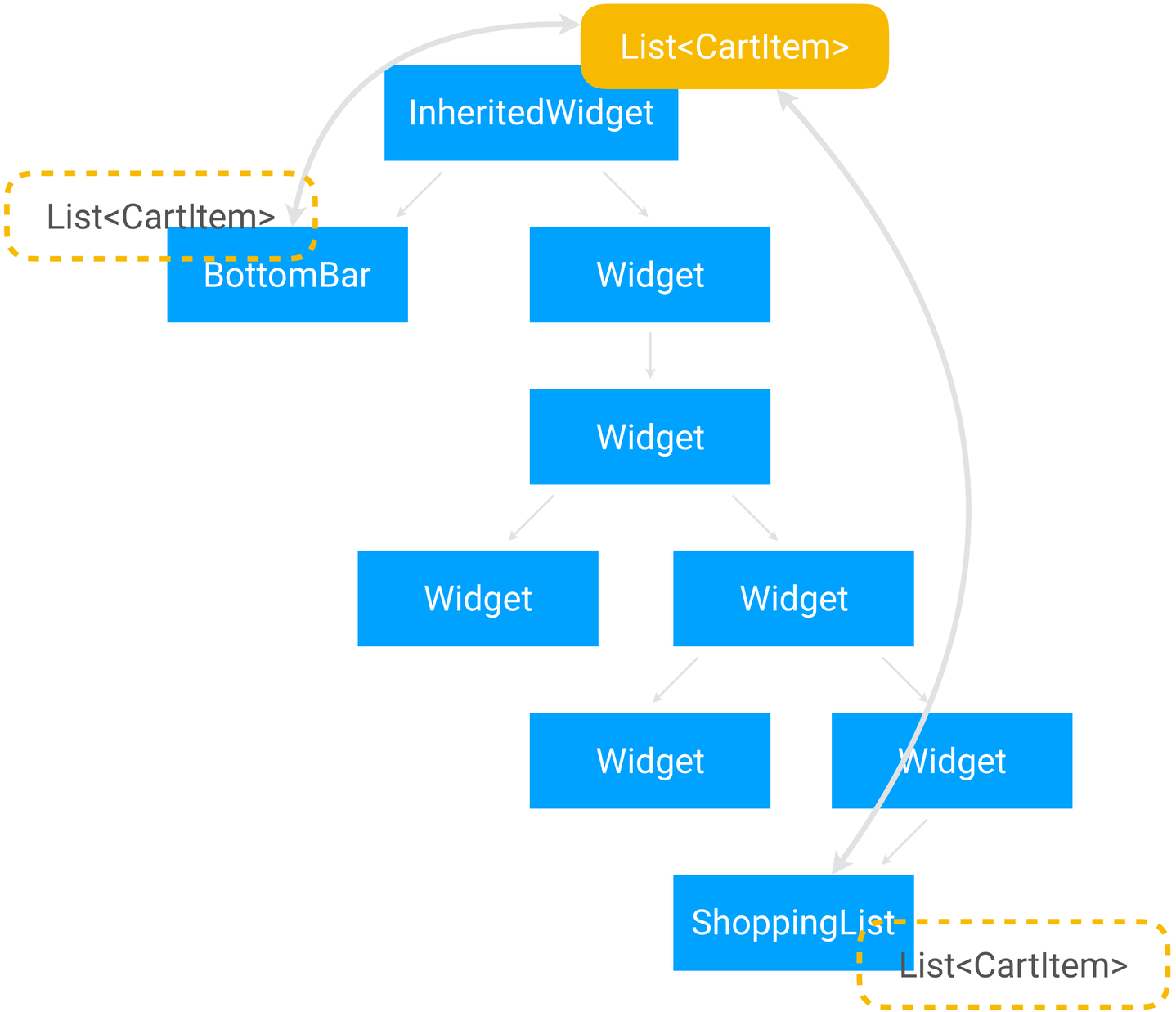
option A
pass data through
all the Widgets 🤮

option B
use solution for
state management 🙌

What are our options?

option B
some solution for **state management** 🙌

Inherited Widget


List<CartItem>
InheritedWidget
List<CartItem>
BottomBar
Widget
Widget
Widget
Widget
Widget
Widget
ShoppingList
List<CartItem>

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

class CartRepository extends InheritedWidget {

}
```

```dart
import 'package:flutter/material.dart';
import 'model/cart_item.dart';

class CartRepository extends InheritedWidget {

  final List<CartItem> cartItems;

}
```

```dart
import 'package:flutter/material.dart';
import 'model/cart_item.dart';

class CartRepository extends InheritedWidget {

  final List<CartItem> cartItems;

  @override
  bool updateShouldNotify(CartRepository oldWidget) {
    return this.cartItems != oldWidget.cartItems;
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'model/cart_item.dart';

class CartRepository extends InheritedWidget {

  final List<CartItem> cartItems;

  @override
  bool updateShouldNotify(CartRepository oldWidget) {
    return this.cartItems != oldWidget.cartItems;
  }

  static CartRepository of(BuildContext context) {
    return context.inheritFromWidgetOfExactType(
        CartRepository) as CartRepository;
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'model/cart_item.dart';

class CartRepository extends InheritedWidget {

  final List<CartItem> cartItems;

  @override
  bool updateShouldNotify(CartRepository oldWidget) {
    return this.cartItems != oldWidget.cartItems;
  }

  static CartRepository of(BuildContext context) {
    return context.inheritFromWidgetOfExactType(
        CartRepository) as CartRepository;
  }
}
```

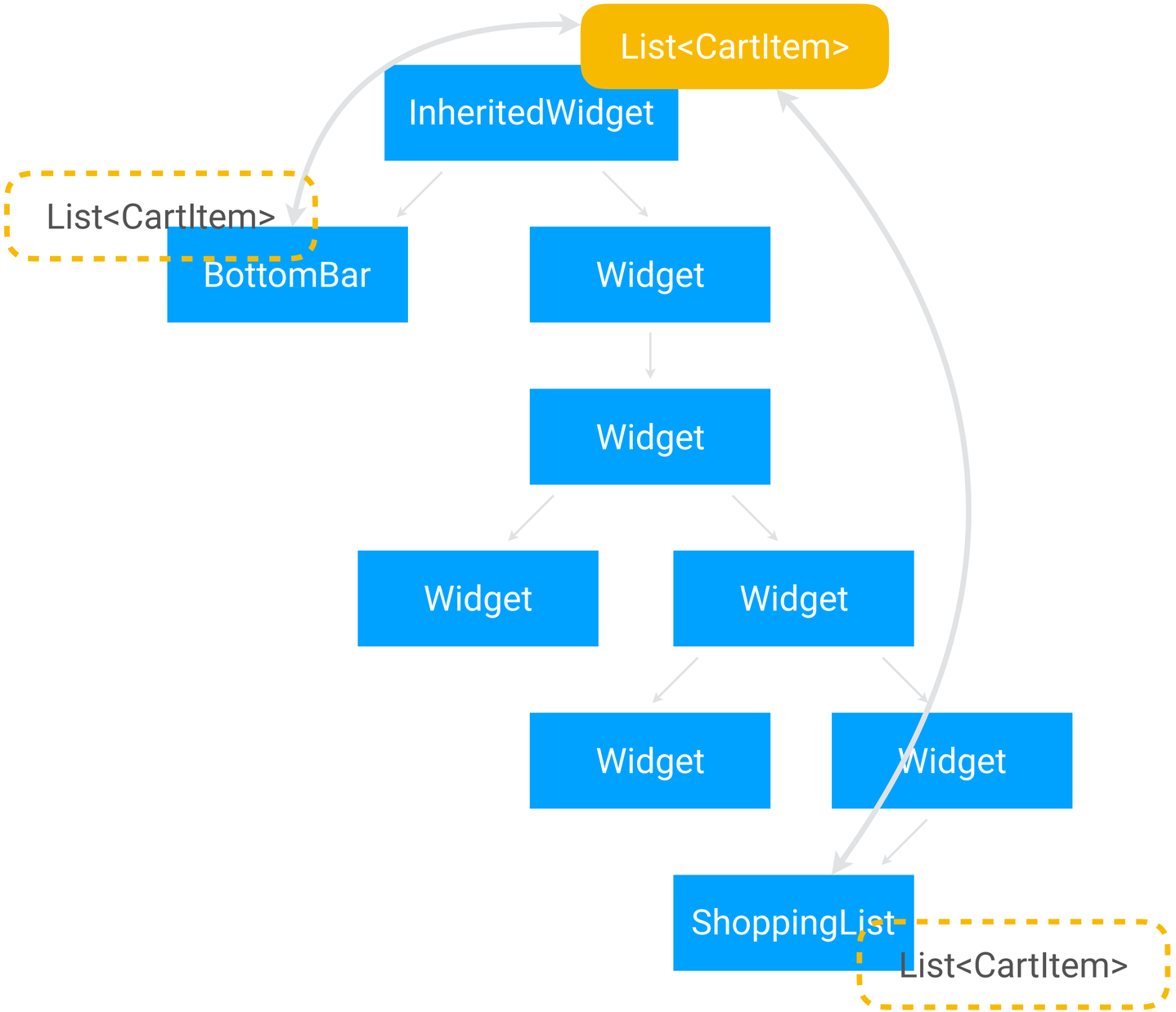
```dart
Theme.of(context).textTheme;

MediaQuery.of(context).size;
```

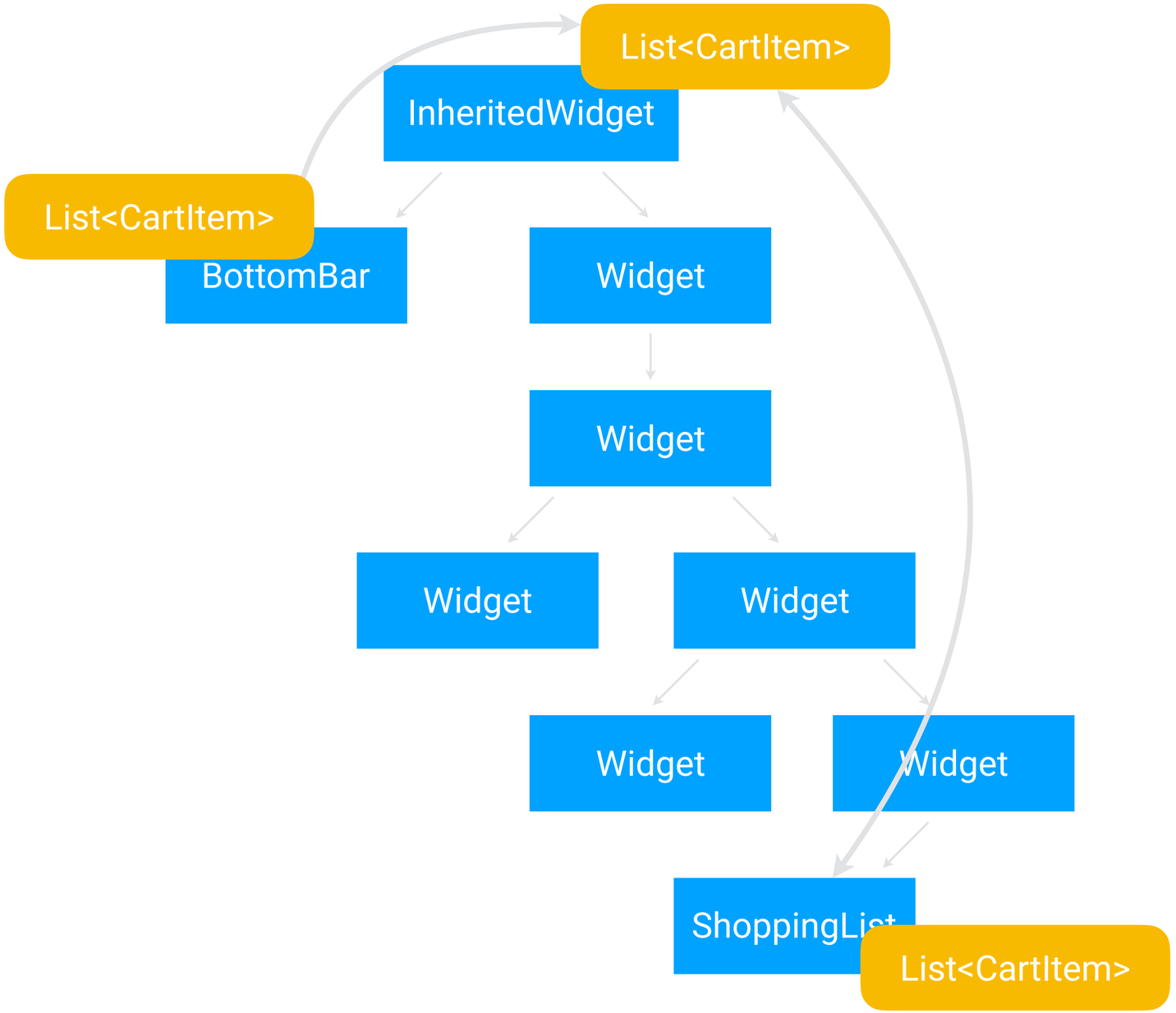
```
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return ShoppingApp(cartItems);
  }
}
```

```dart
class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return CartRepository(
      cartItems: cartItems,
      child: ShoppingApp(cartItems),
    );
  }
}
```

```
class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: cartItems.length,
      itemBuilder: (context, position) =>
          ShoppingListItem(cartItems[position]),
    );
  }
}
```

```
class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {

    var cartItems = CartRepository.of(context).cartItems;

    return ListView.builder(
      itemCount: cartItems.length,
      itemBuilder: (context, position) =>
          ShoppingListItem(cartItems[position]),
    );
  }
}
```


List<CartItem>
InheritedWidget
List<CartItem>
BottomBar
Widget
Widget
Widget
Widget
Widget
Widget
ShoppingList
List<CartItem>


List<CartItem>
InheritedWidget
List<CartItem>
BottomBar
Widget
Widget
Widget
Widget
Widget
Widget
ShoppingList
List<CartItem>

What are our options?

option B
some solution for **state management** 🙌

Inherited Widget
ScopedModel
BLoC
Redux

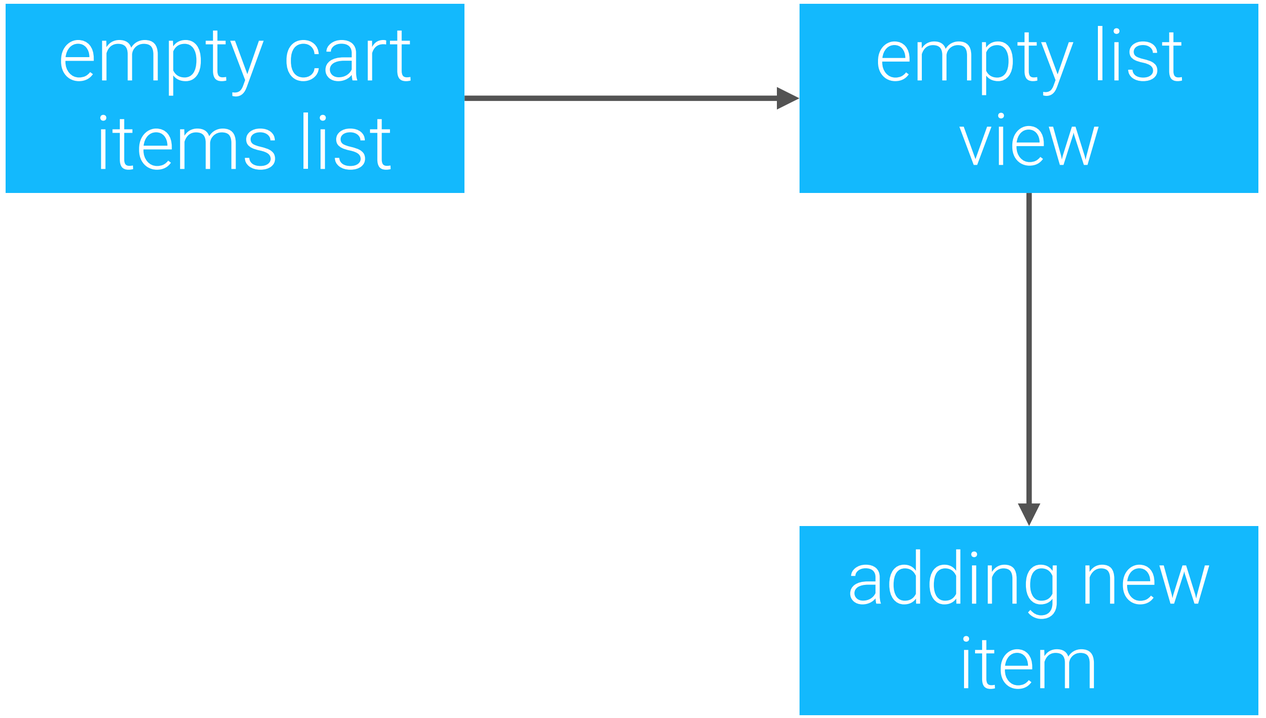
Redux

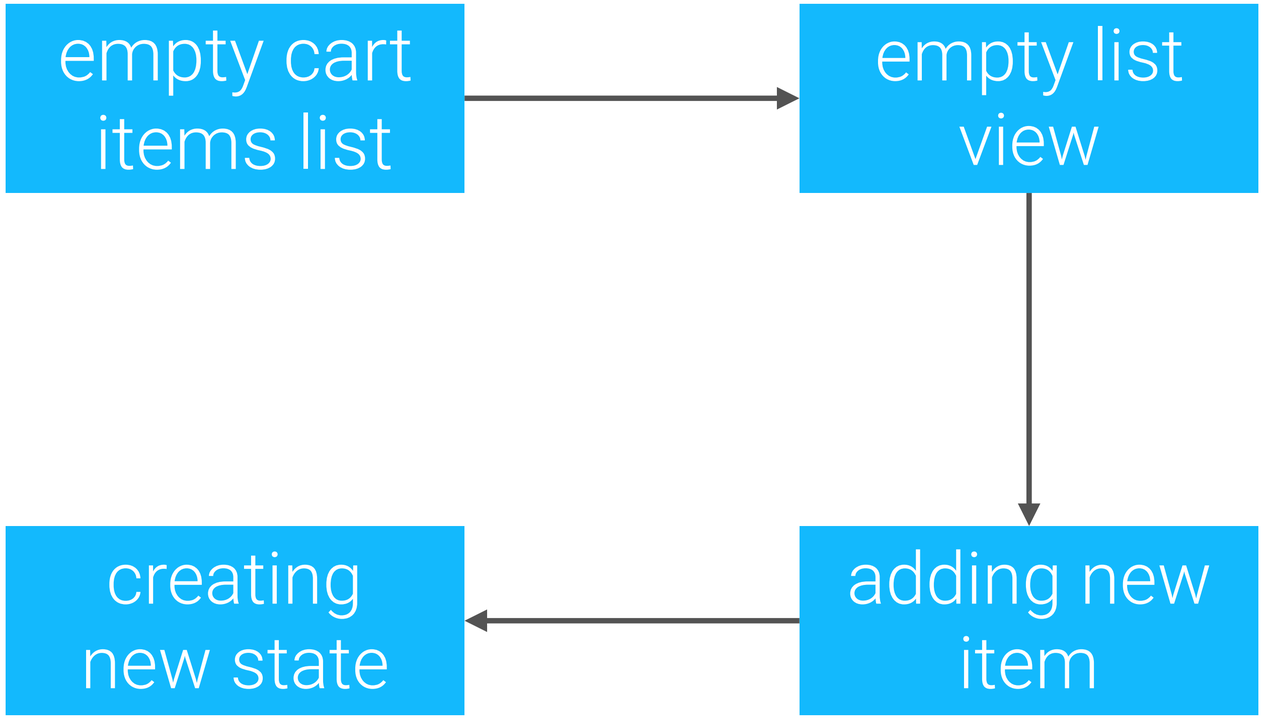
All about **State**

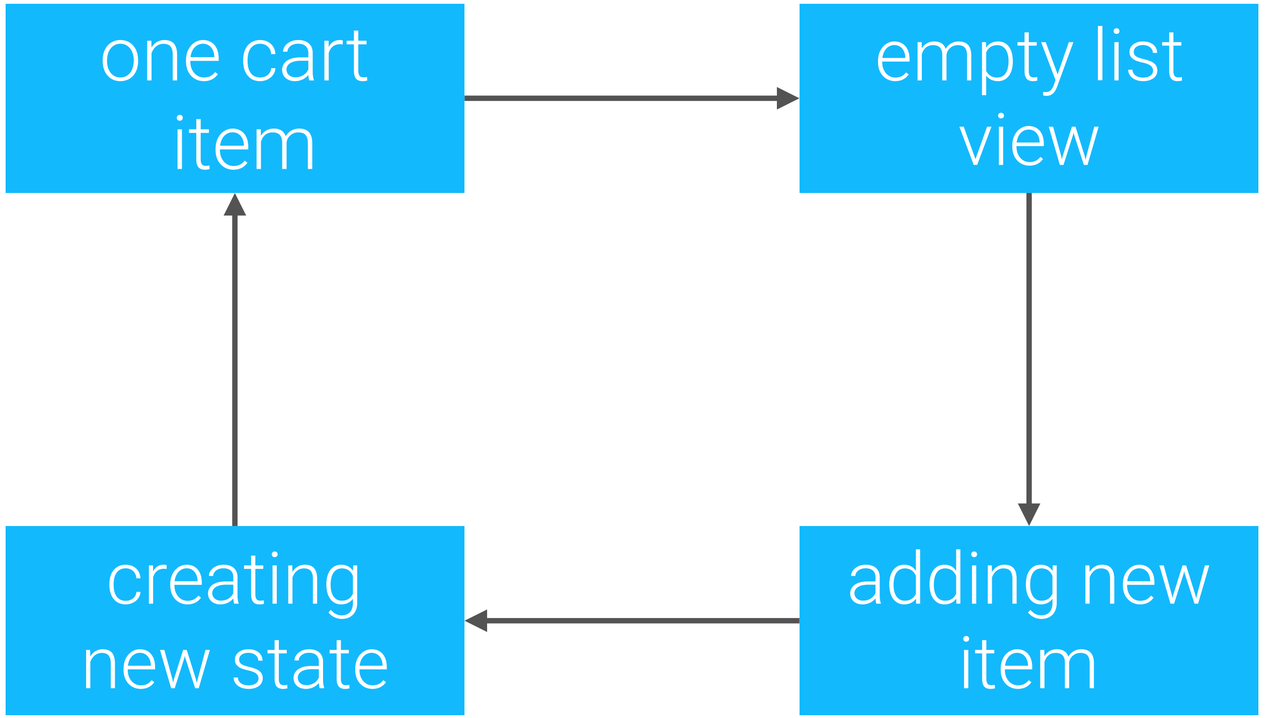
UI = f(state)

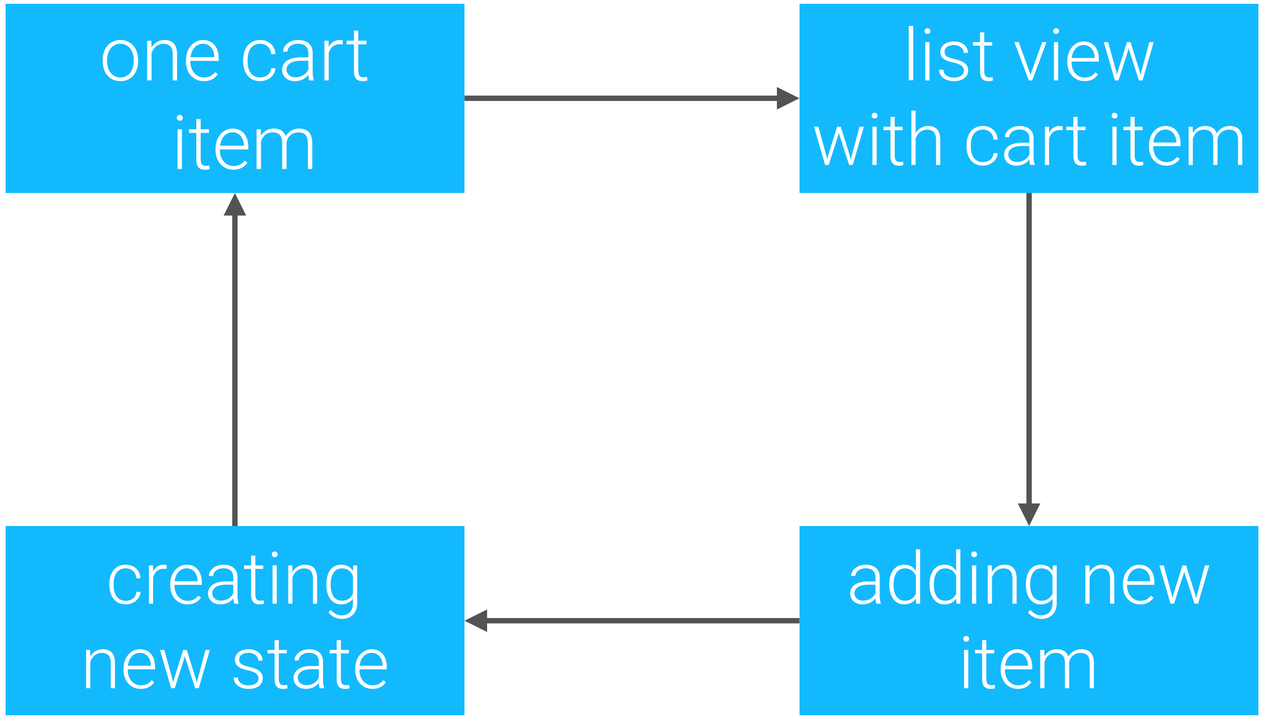
empty cart
items list

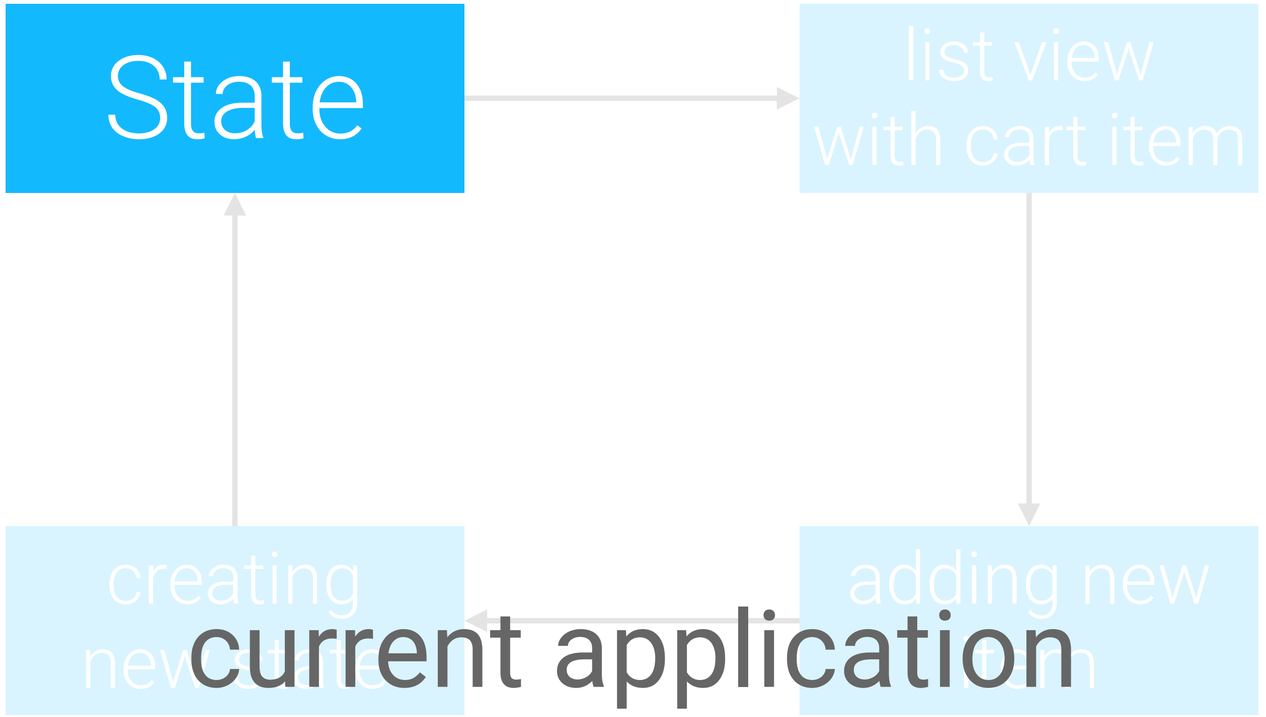

empty cart
items list
empty list
view


empty cart
items list
empty list
view
adding new
item


empty cart
items list
empty list
view
adding new
item
creating
new state


one cart item
empty list view
adding new item
creating new state


one cart item
list view with cart item
adding new item
creating new state



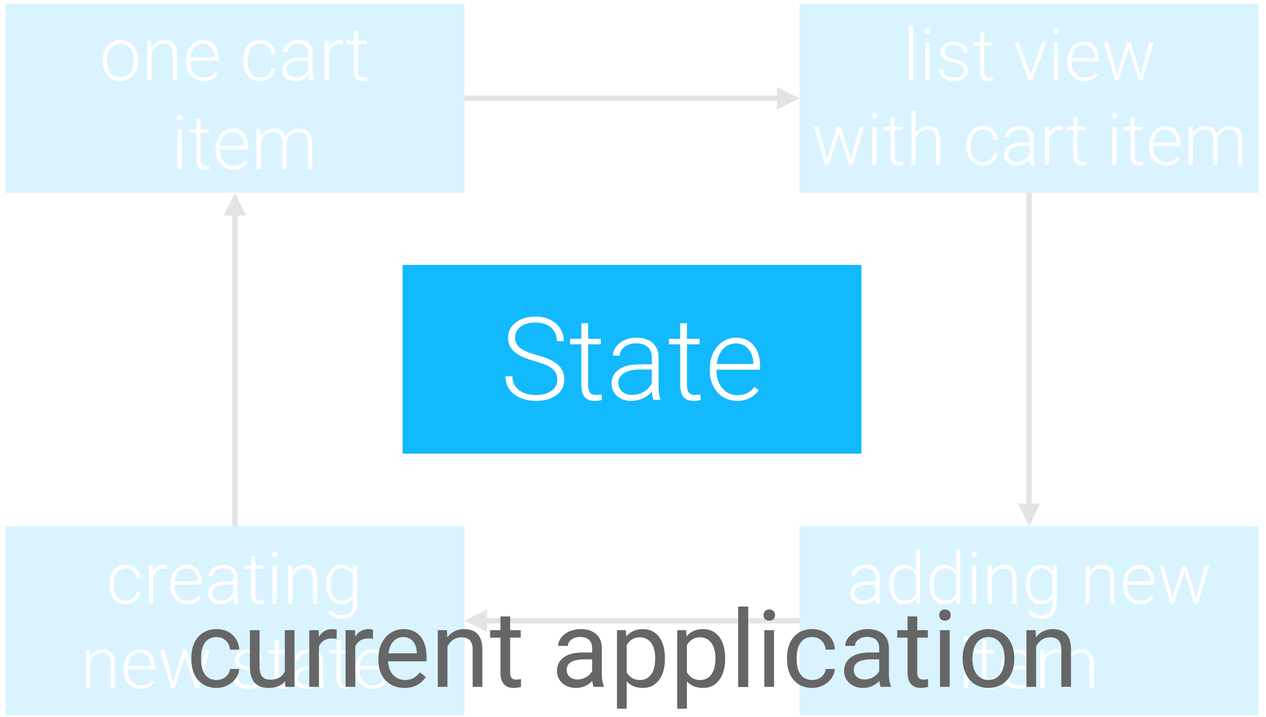
current application state

State

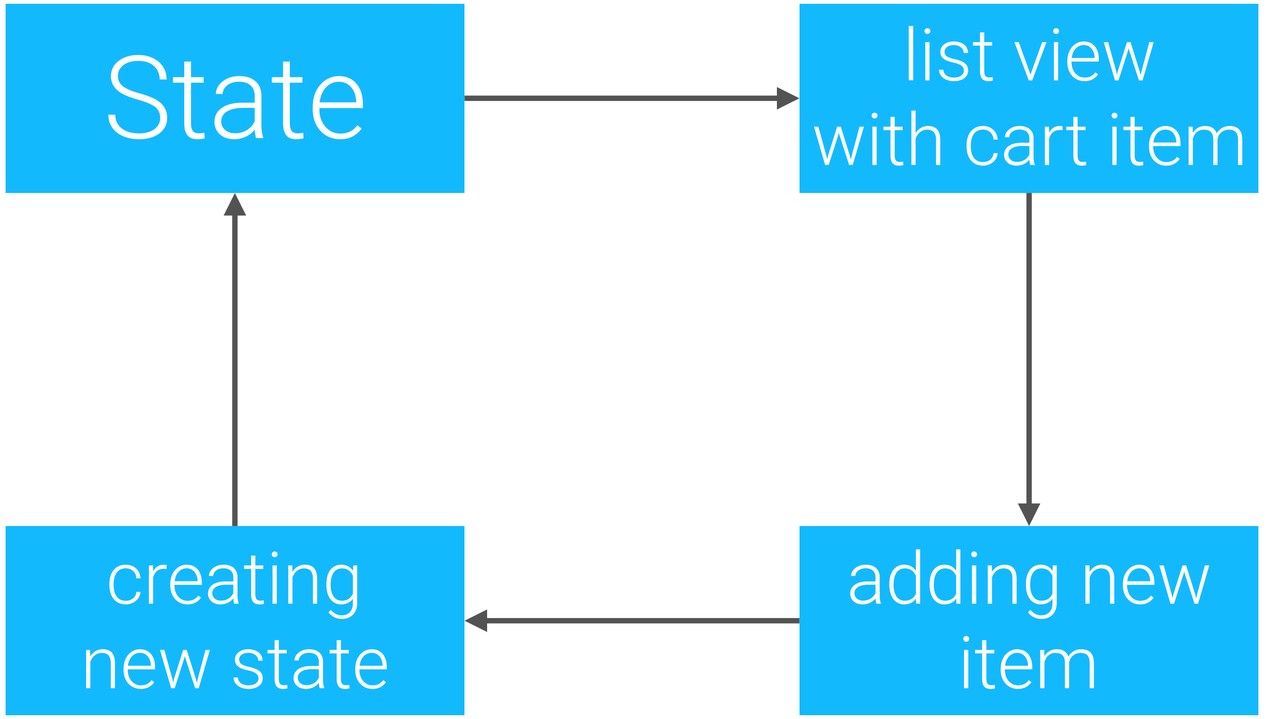
list of users
details of product
number of followers

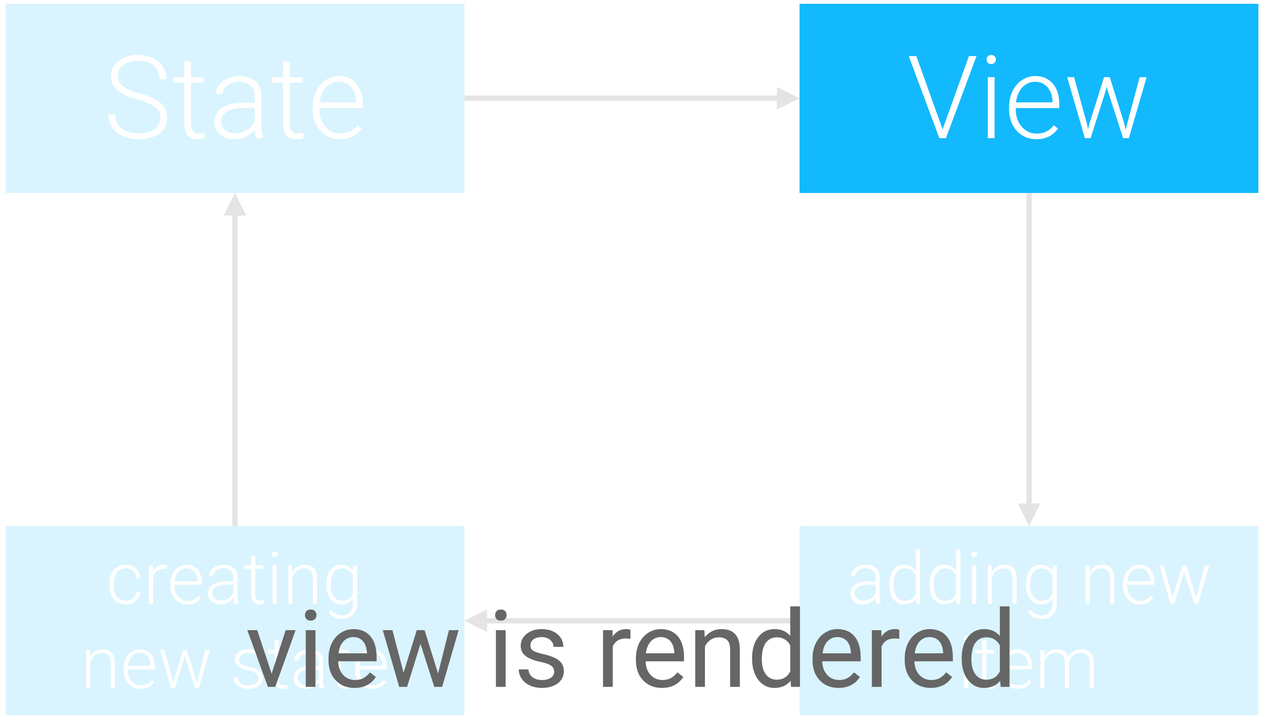
single source of truth

```dart
class AppState {
  final List<CartItem> cartItems;
  AppState(this.cartItems);
}
```



current application state


State
list view
with cart item
adding new
item
creating
new state



view is rendered based on state

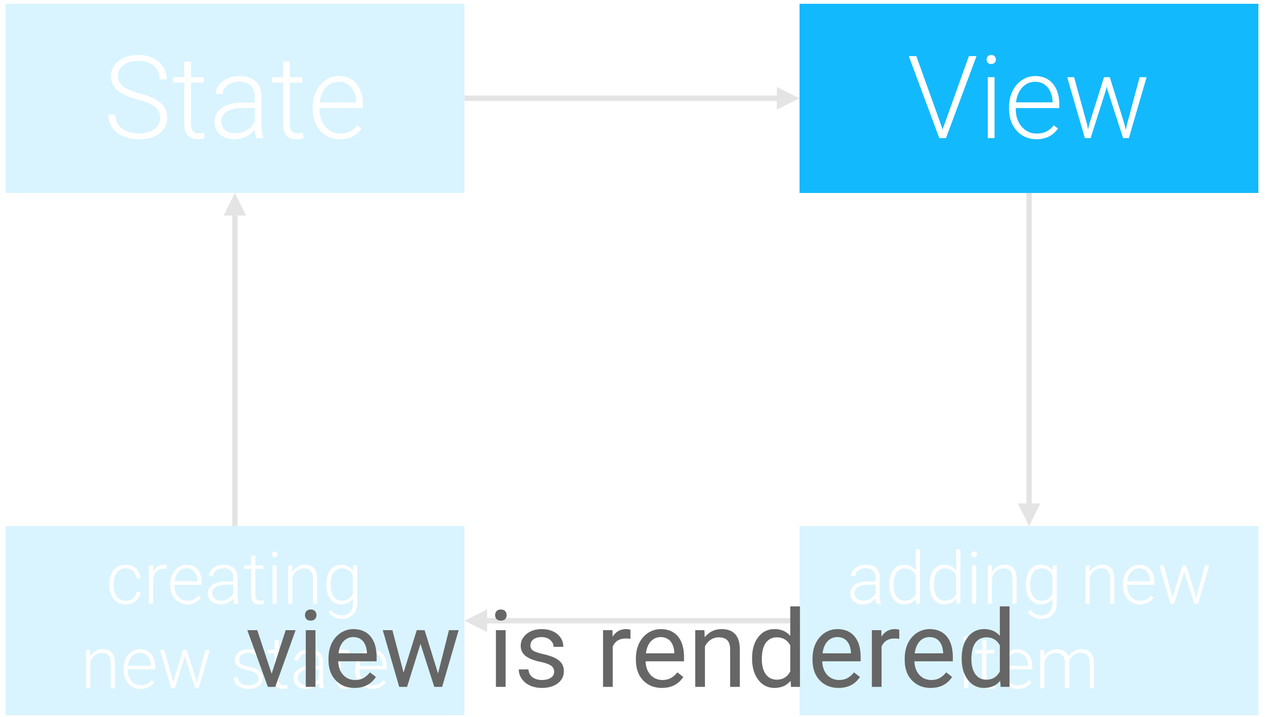
```dart
import 'package:flutter_redux/flutter_redux.dart';

class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {

  }
}
```

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

class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, List<CartItem>>(

    );
  }
}
```

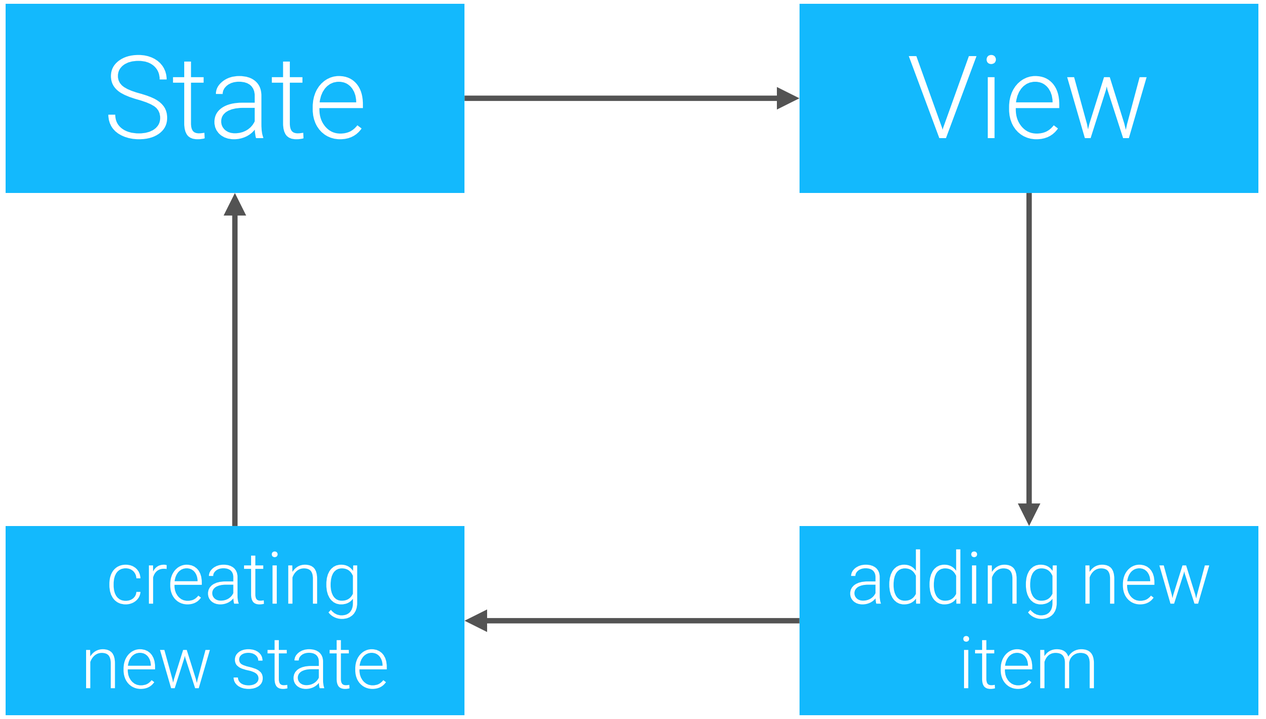
```dart
import 'package:flutter_redux/flutter_redux.dart';

class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, List<CartItem>>(
      converter: (store) => store.state.cartItems,

    );
  }
}
```

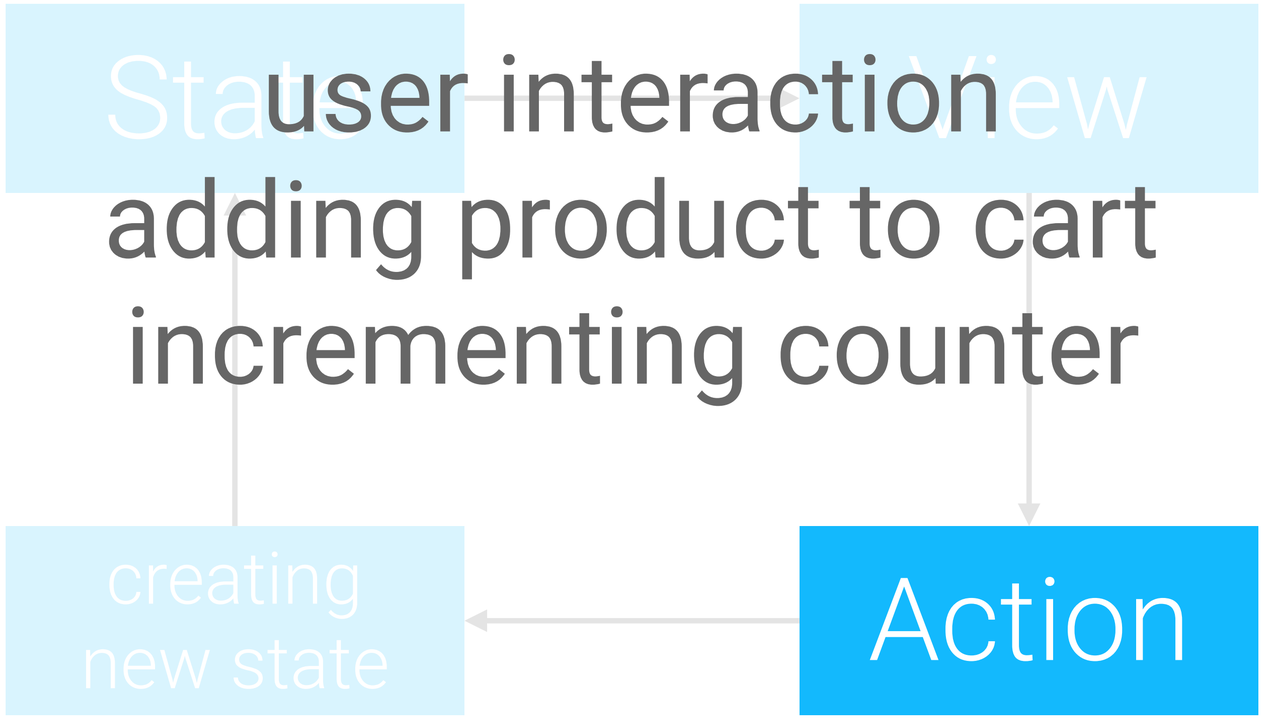
```dart
import 'package:flutter_redux/flutter_redux.dart';

class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, List<CartItem>>(
      converter: (store) => store.state.cartItems,
      builder: (context, cartItems) {

      },
    );
  }
}
```

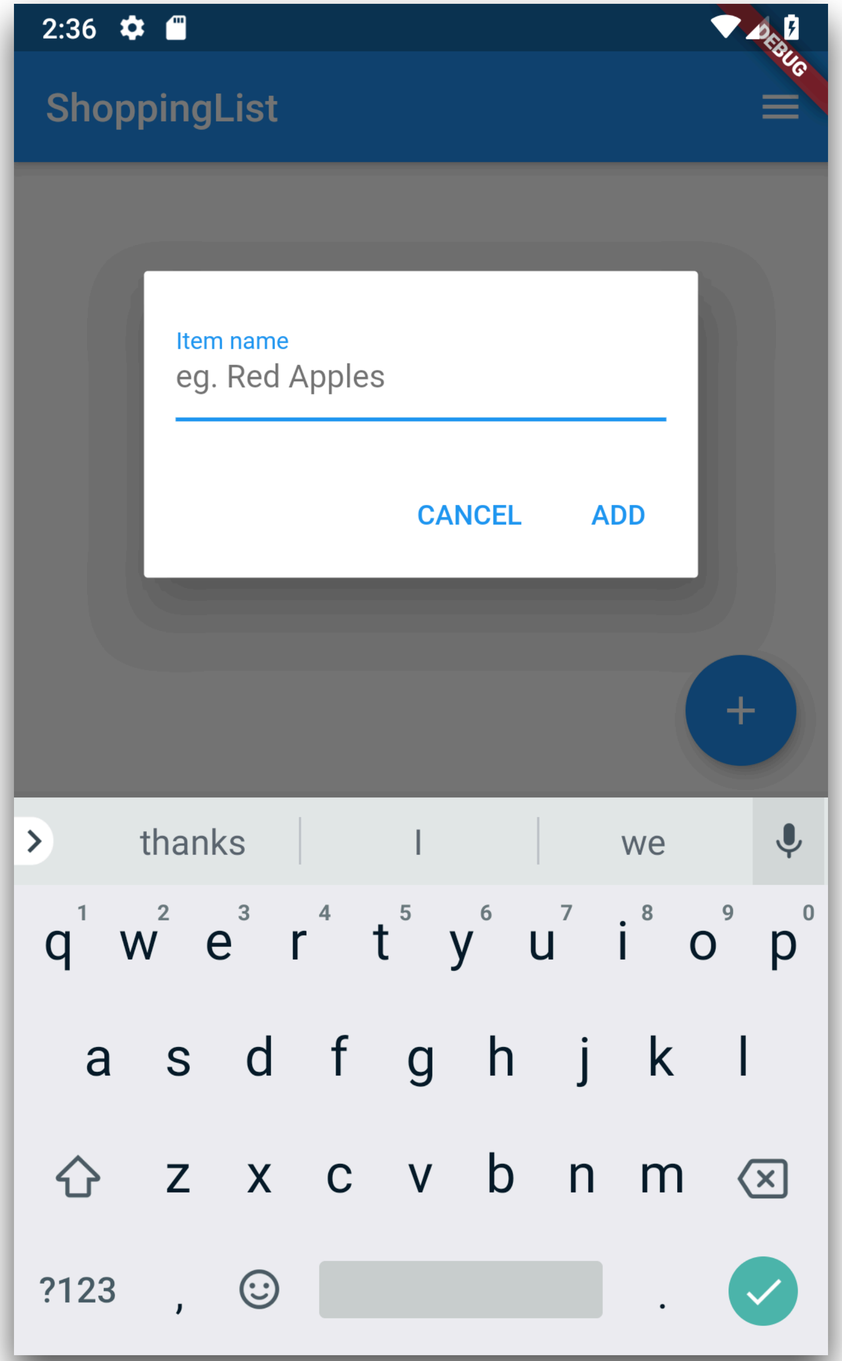
```dart
import 'package:flutter_redux/flutter_redux.dart';

class ShoppingList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, List<CartItem>>(
      converter: (store) => store.state.cartItems,
      builder: (context, cartItems) {
        return ListView.builder(
          itemCount: cartItems.length,
          itemBuilder: (context, position) =>
              ShoppingListItem(cartItems[position]),
        );
      },
    );
  }
}
```



view is rendered based on state


State
View
adding new item
creating new state


State
View
user interaction
adding product to cart
incrementing counter
creating
new state
Action


2:36
DEBUG
ShoppingList
Item name
eg. Red Apples
CANCEL
ADD
thanks
I
we
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
,
.

```dart
class AddItemAction {
  final CartItem item;

  AddItemAction(this.item);
}
```

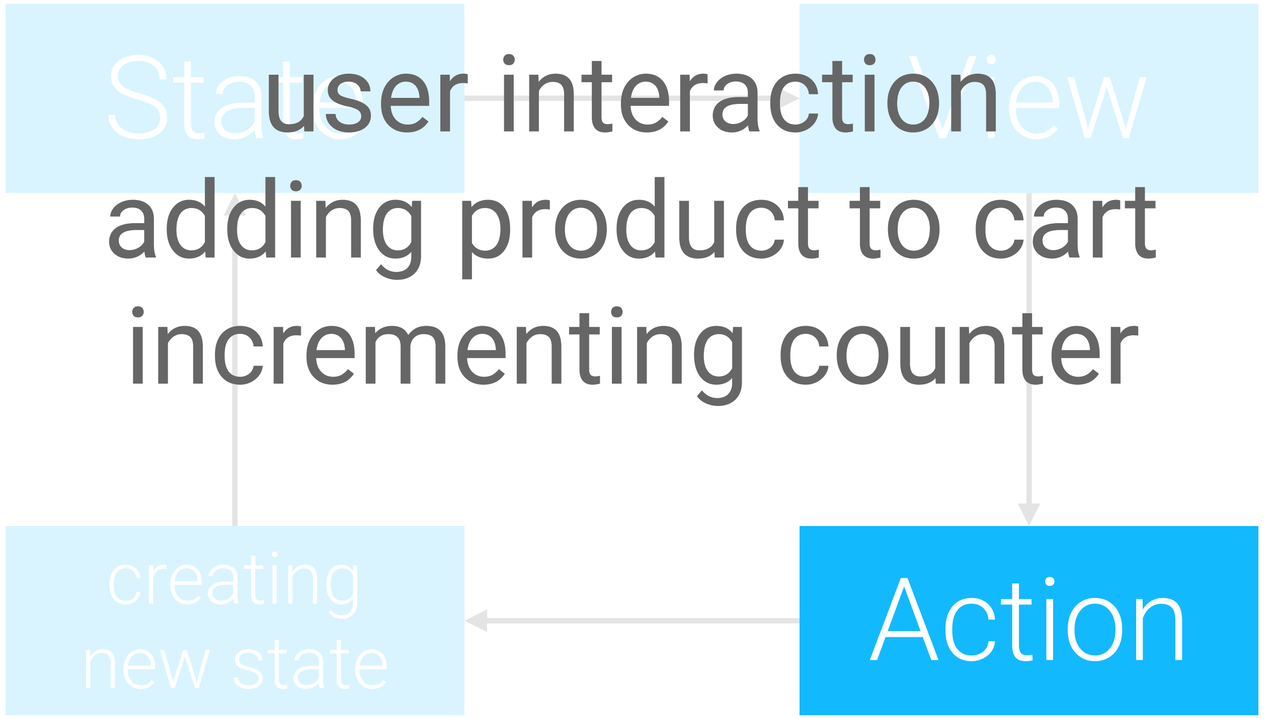
```dart
class AddItemDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {

  }
}
```

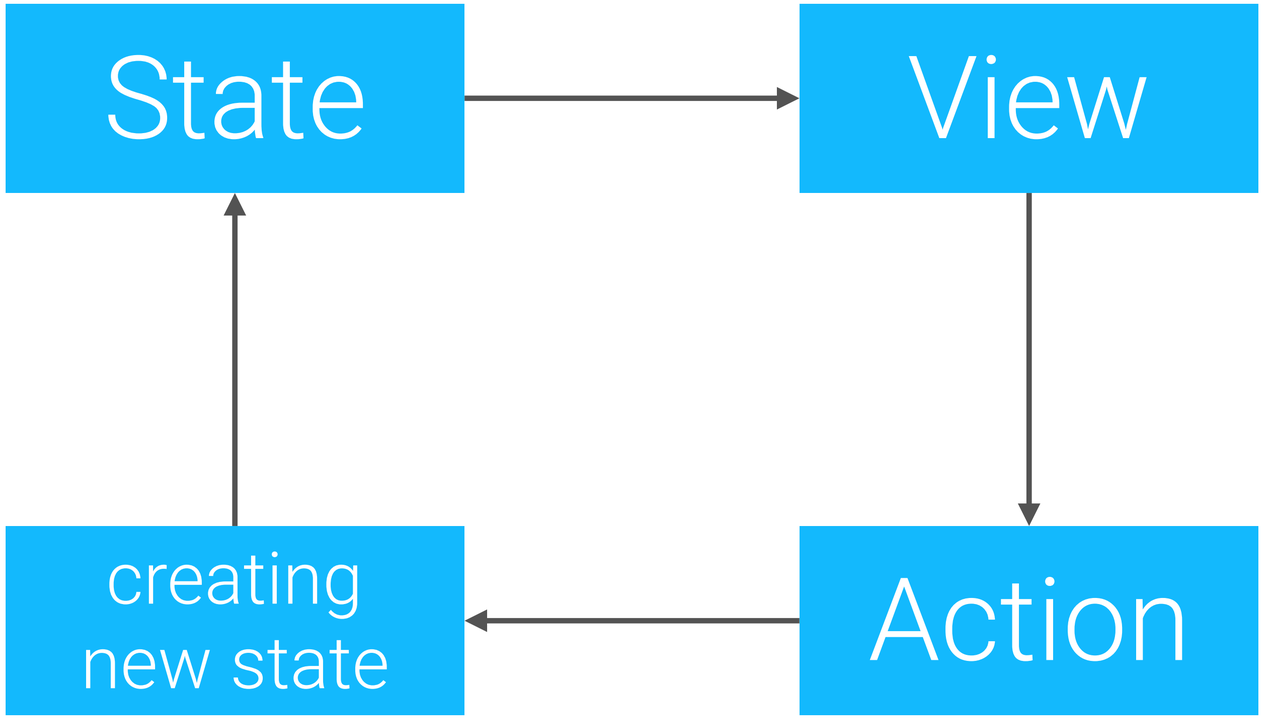
```
class AddItemDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, Function(String)>(

    );
  }
}
```

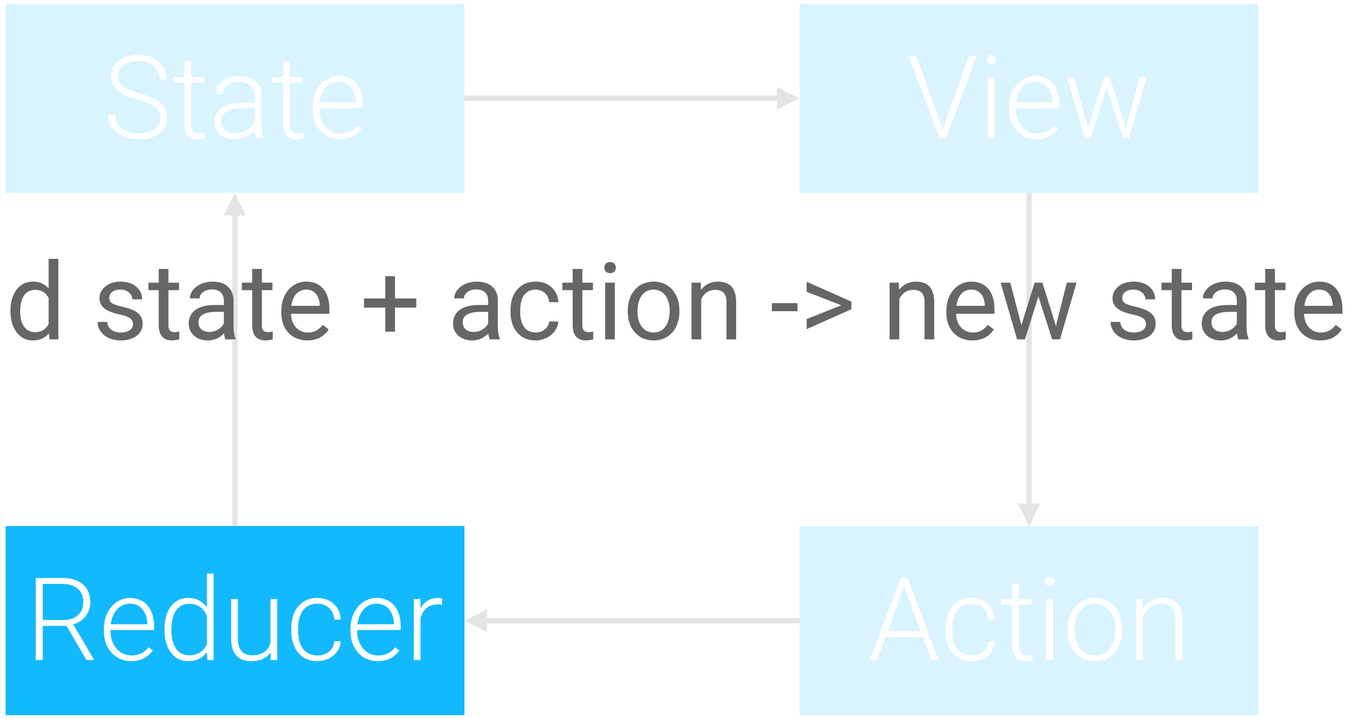
```
class AddItemDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, Function(String)>(
      converter: (store) {
        return (itemName) => store.dispatch(
            AddItemAction(CartItem(itemName));
      },
    );
  }
}
```

```
class AddItemDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, Function(String)>(
      converter: (store) {
        return (itemName) => store.dispatch(
            AddItemAction(CartItem(itemName));
      },
      builder: (context, callback) {
        return AddItemDialogWidget(callback);
      },
    );
  }
}
```

```
class AddItemDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StoreConnector<AppState, Function(String)>(
      converter: (store) {
        return (itemName) => store.dispatch(
            AddItemAction(CartItem(itemName));
      },
      builder: (context, callback) {
        return AddItemDialogWidget(callback);
      },
    );
  }
}
```


State
View
user interaction
adding product to cart
incrementing counter
creating
new state
Action


State
View
creating
new state
Action


State
View
old state + action -> new state
Reducer
Action

Reducer

old state + action -> new state

```
AppState addItem(AppState oldState, AddItemAction action) {
  return AppState(
      List.from(oldState.cartItems)..add(action.item)
  );
}
```

Reducer

old state + action -> new state

```
AppState addItem(AppState oldState, AddItemAction action) {
  return AppState(
      List.from(oldState.cartItems)..add(action.item)
  );
}
```

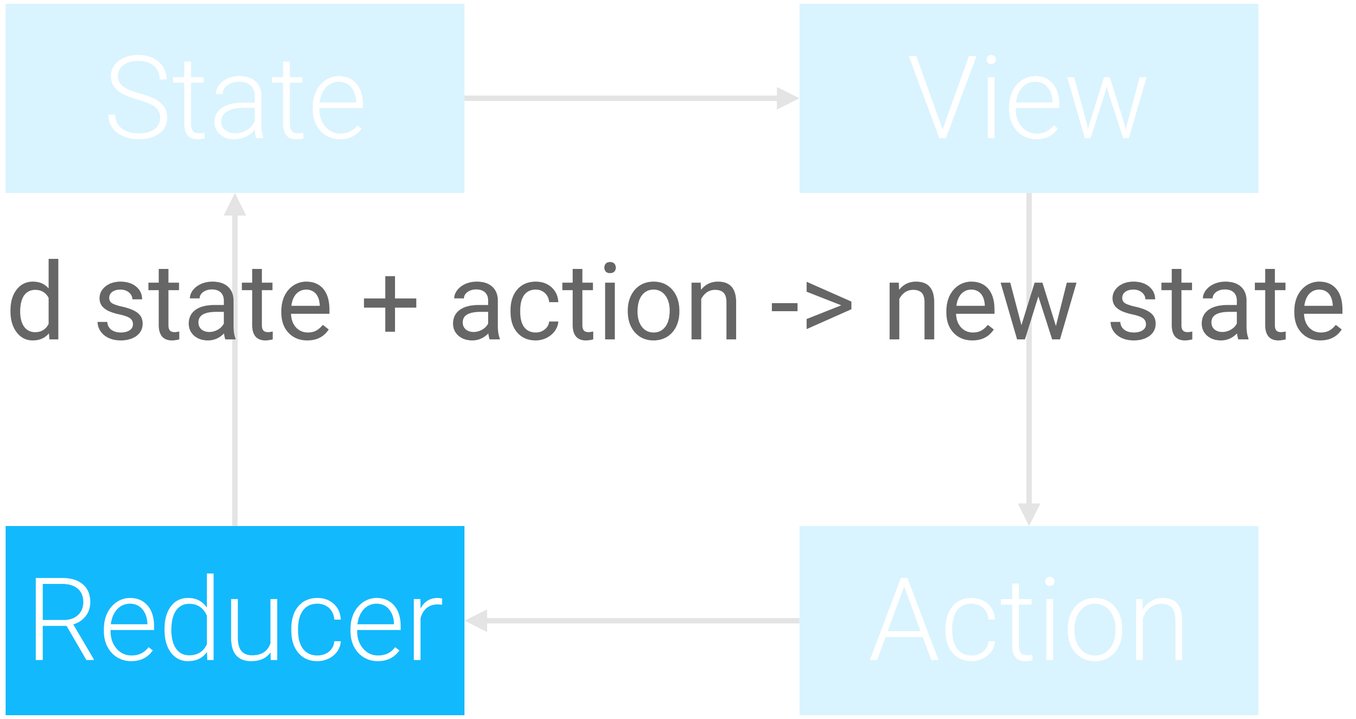
state is read-only

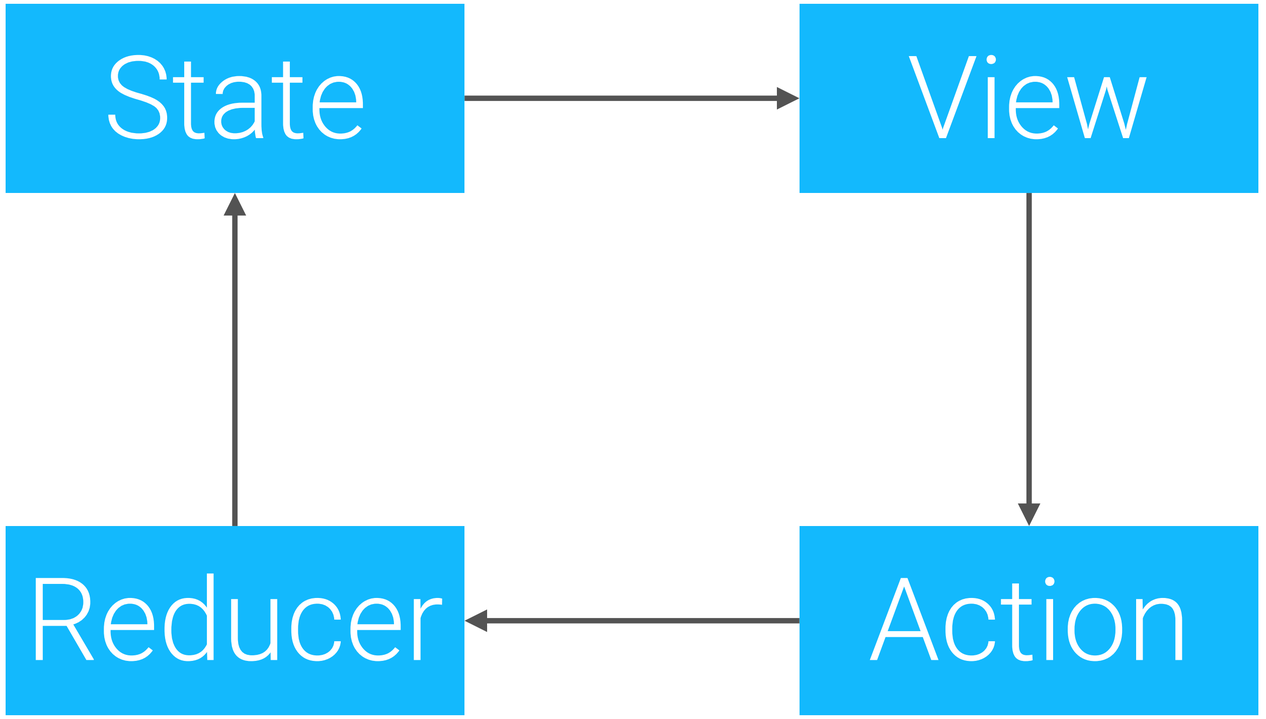
Reducer

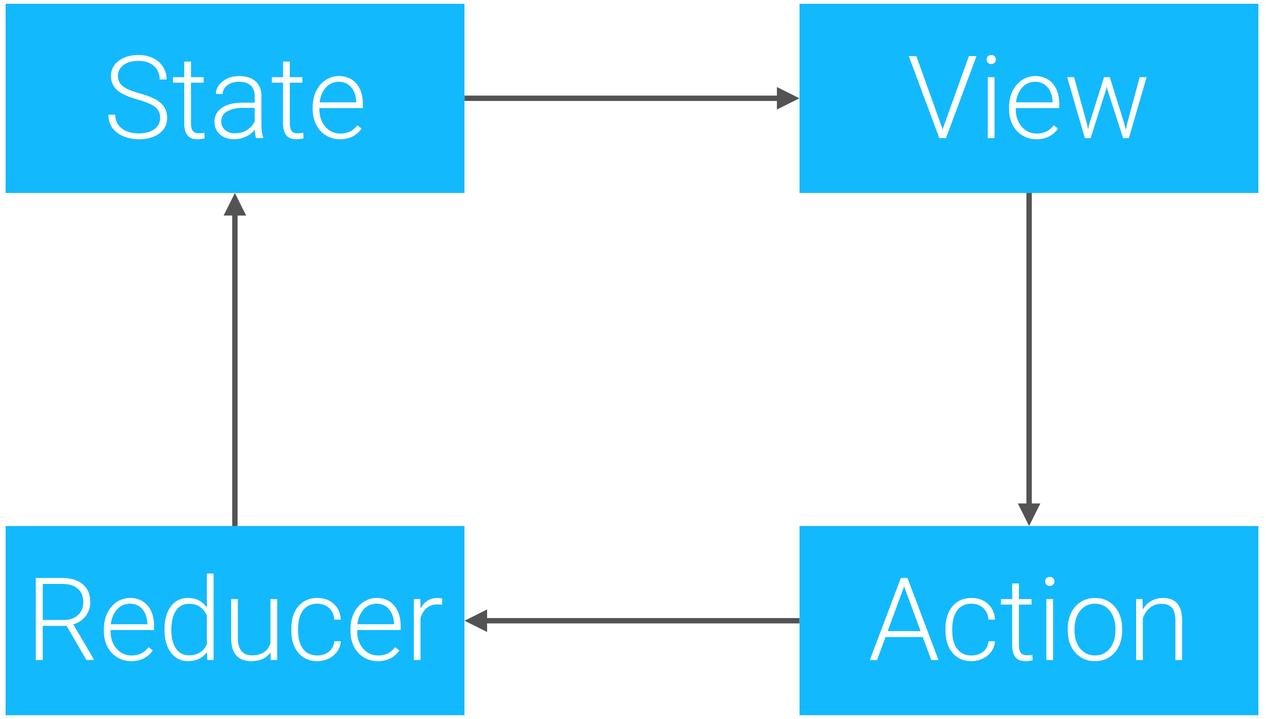
old state + action -> new state

```
AppState addItem(AppState oldState, AddItemAction action) {
  return AppState(
      List.from(oldState.cartItems)..add(action.item)
  );
}
```

state is read-only
reducers are pure functions


State
View
old state + action -> new state
Reducer
Action


State
View
Reducer
Action

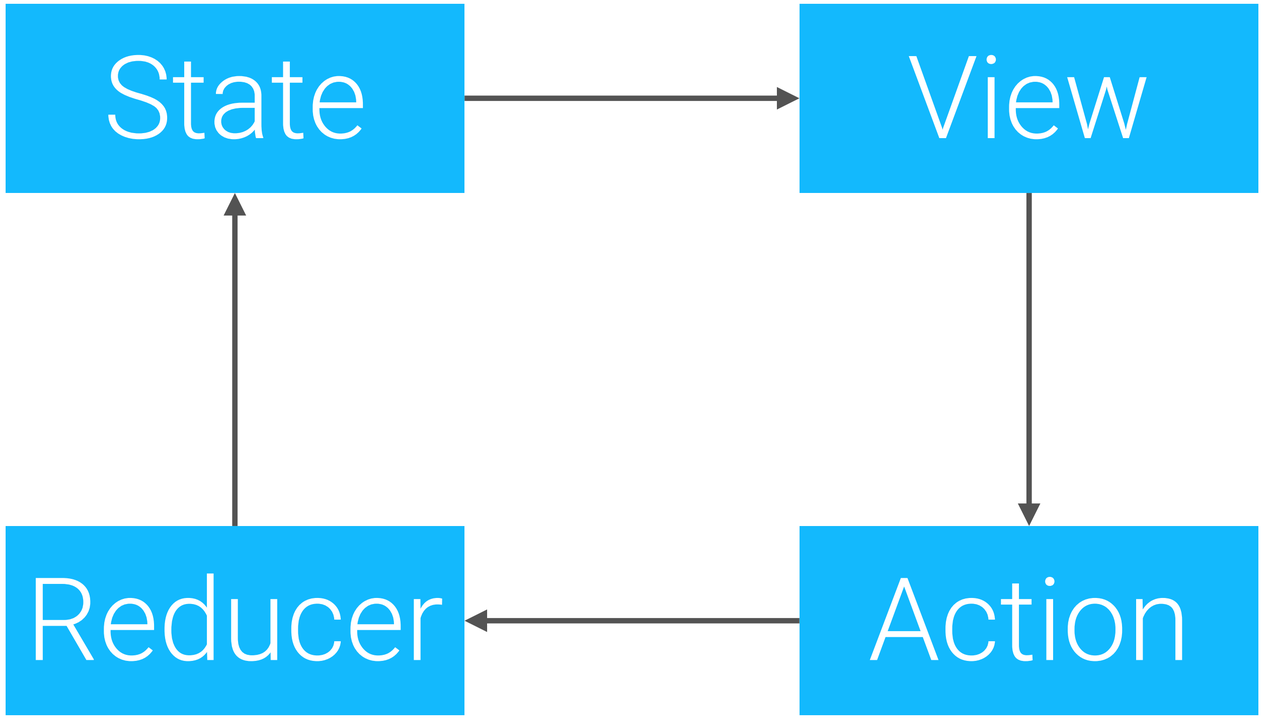


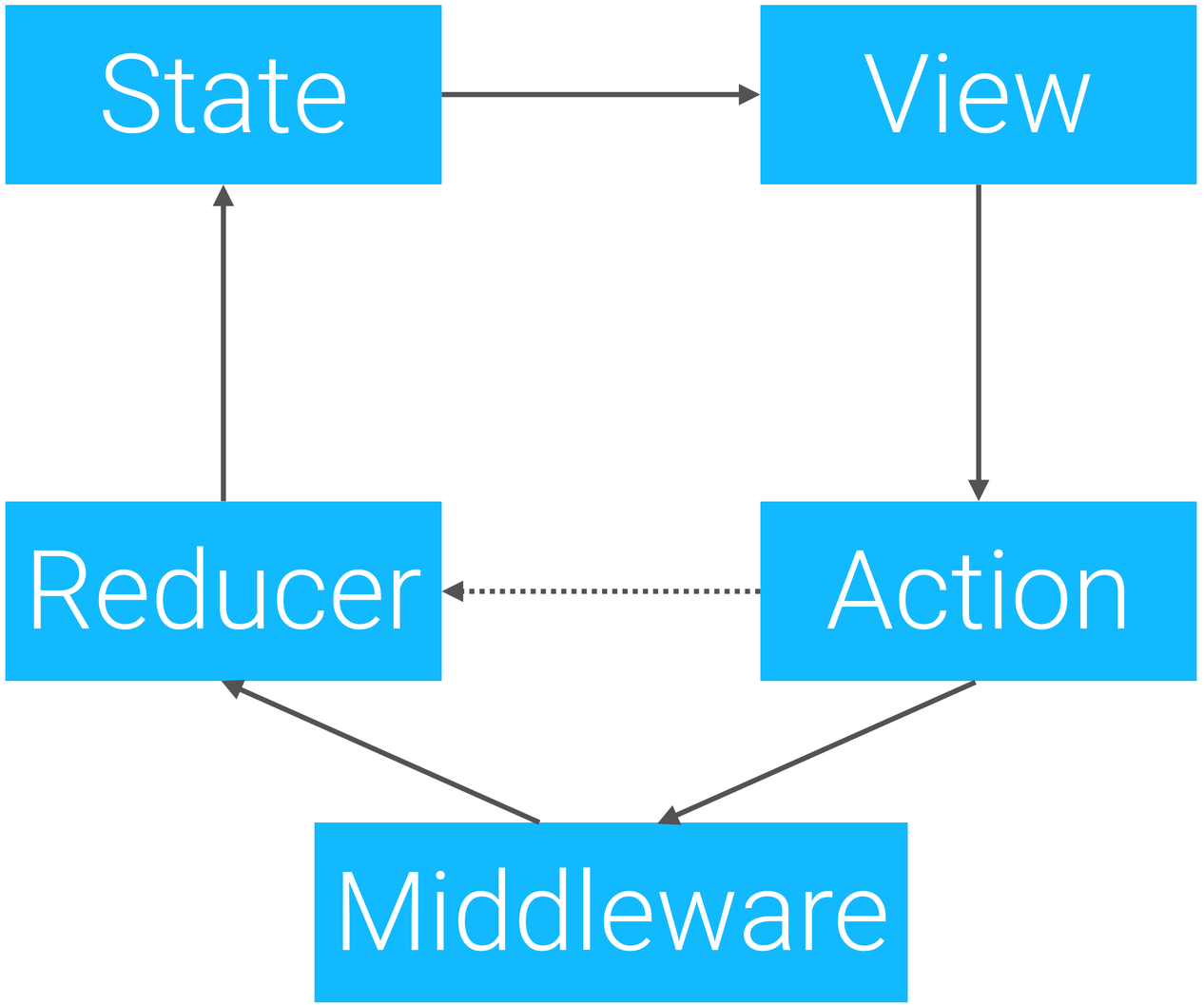
unidirectional

All about **State**

UI = f(state)

What about **complex examples**?


State
View
Reducer
Action


State
View
Reducer
Action
Middleware

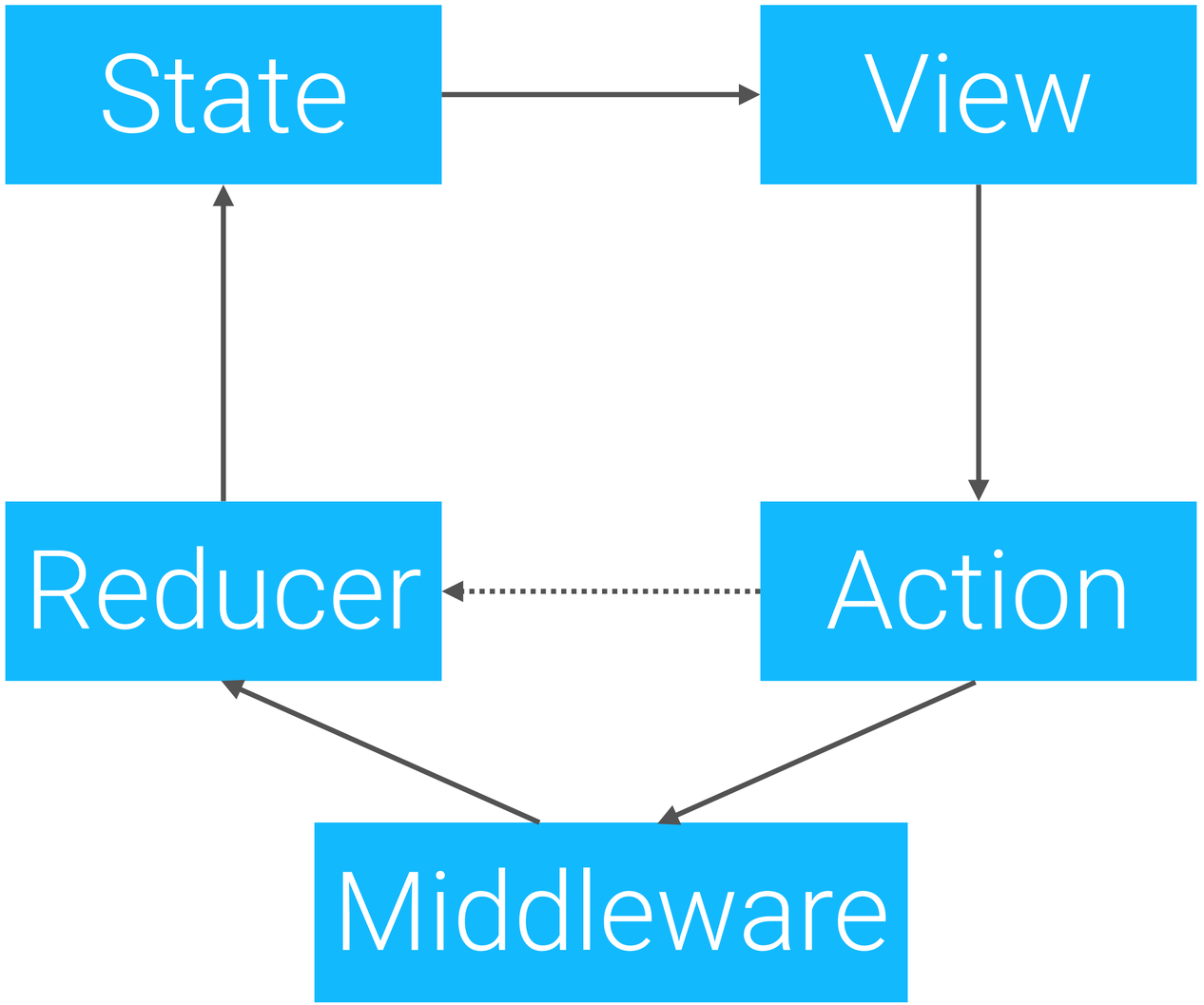
```
class ApiMiddleware extends MiddlewareClass<AppState> {

  @override
  void call(
      Store<AppState> store, action, NextDispatcher next) {

  }
}
```

```dart
class ApiMiddleware extends MiddlewareClass<AppState> {

  @override
  void call(
      Store<AppState> store, action, NextDispatcher next) {

    if (action is FetchCartItems) {

    }

  }
}
```

```dart
class ApiMiddleware extends MiddlewareClass<AppState> {

  @override
  void call(
      Store<AppState> store, action, NextDispatcher next) {

    if (action is FetchCartItems) {
      var cartItems = apiClient.fetchCartItems();
    }

  }
}
```

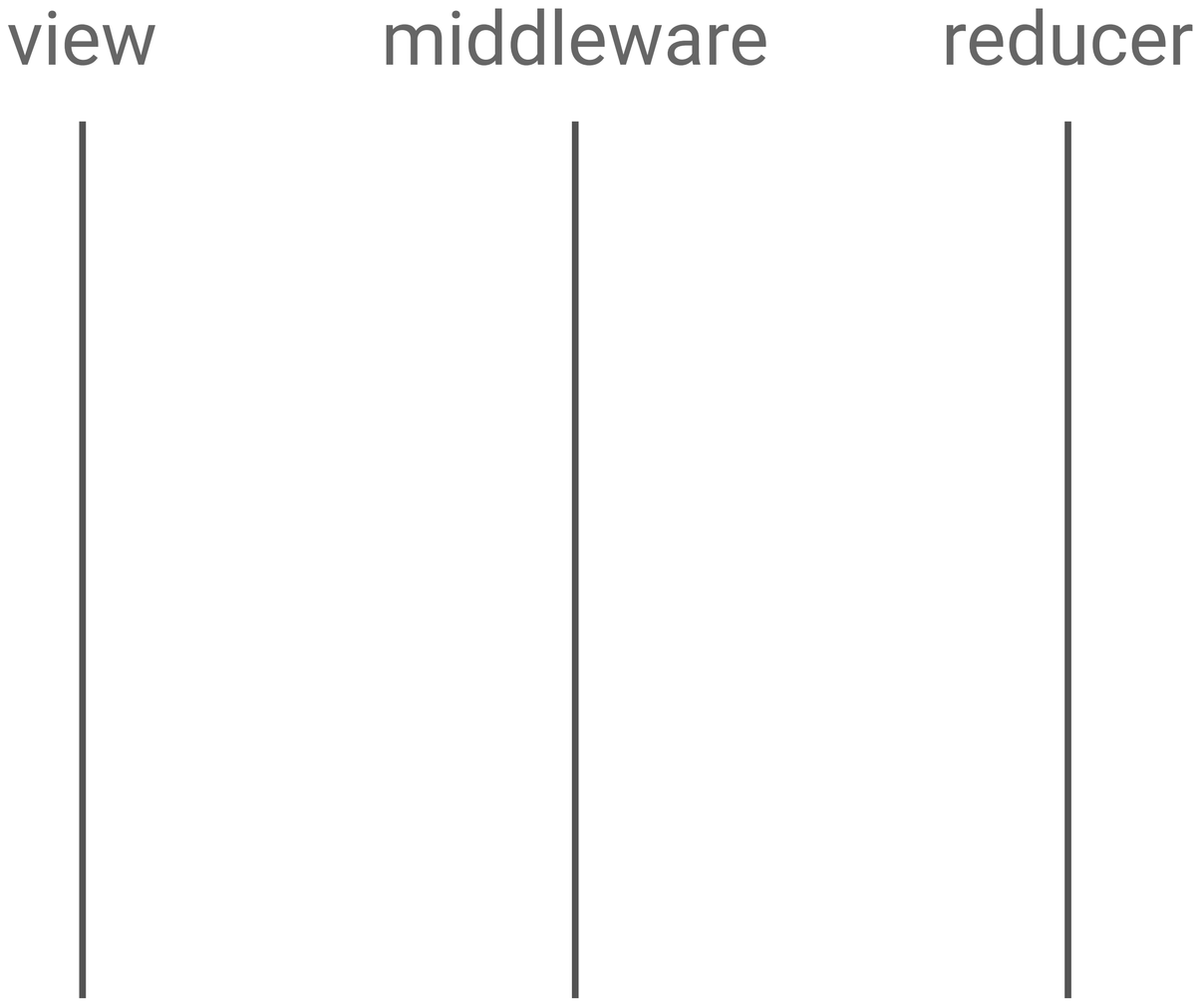
```dart
class ApiMiddleware extends MiddlewareClass<AppState> {

  @override
  void call(
      Store<AppState> store, action, NextDispatcher next) {

    if (action is FetchCartItems) {
      var cartItems = apiClient.fetchCartItems();
      store.dispatch(CartItemsFetched(cartItems));
    }

  }
}
```

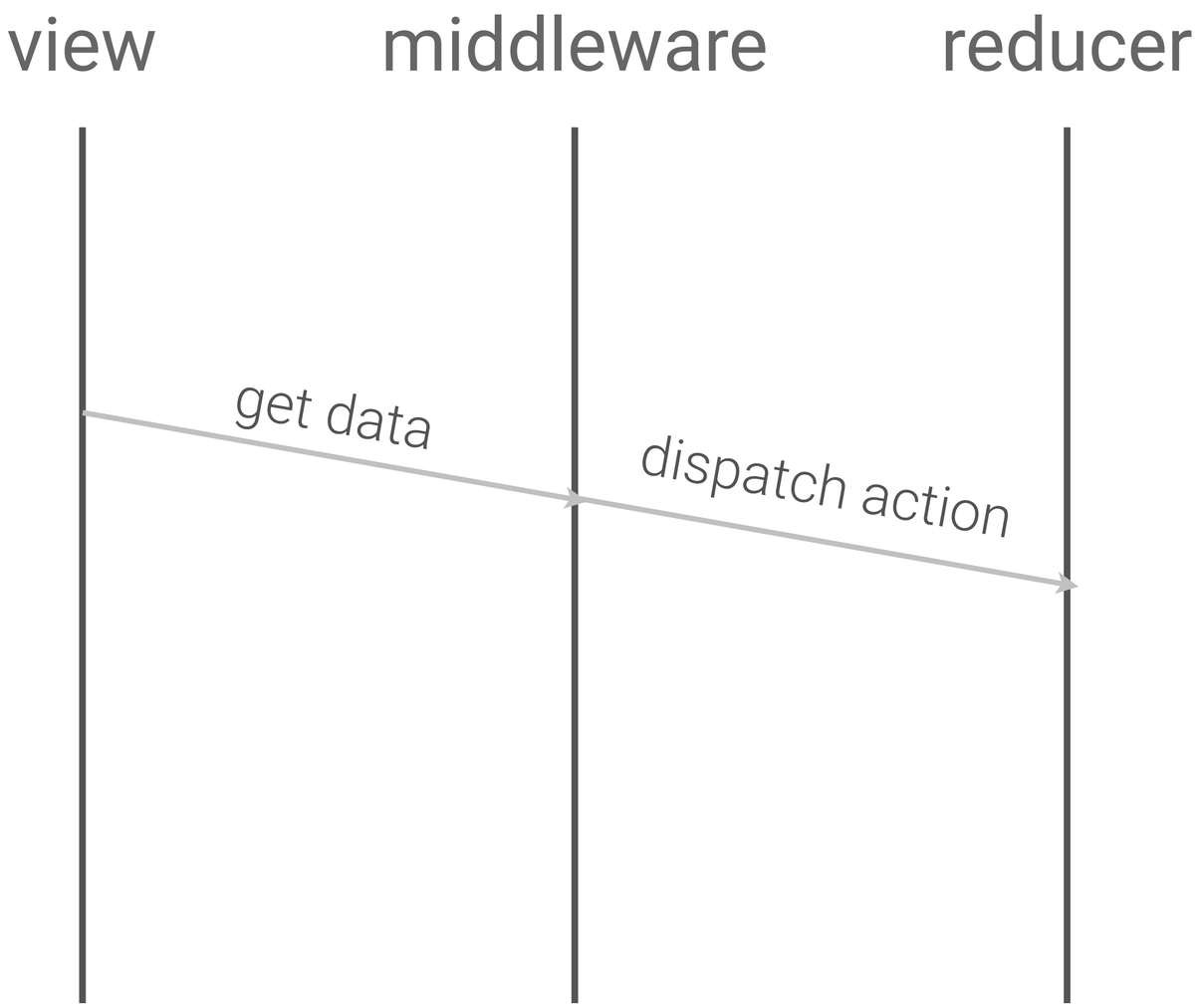
```
AppState loadItems(AppState state, CartItemsFetched action) {
  return AppState(action.cartItems);
}
```

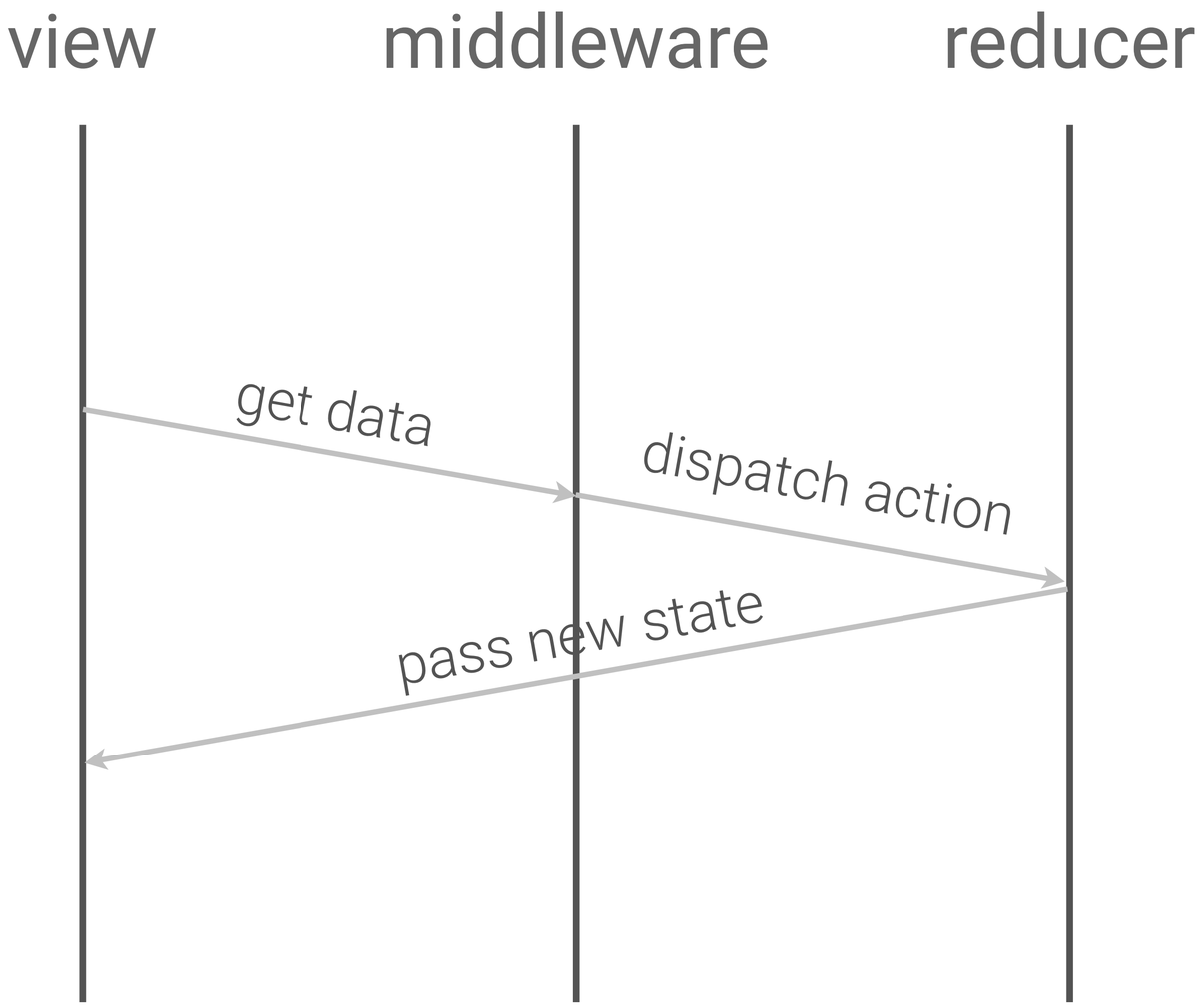

State
View
Reducer
Action
Middleware

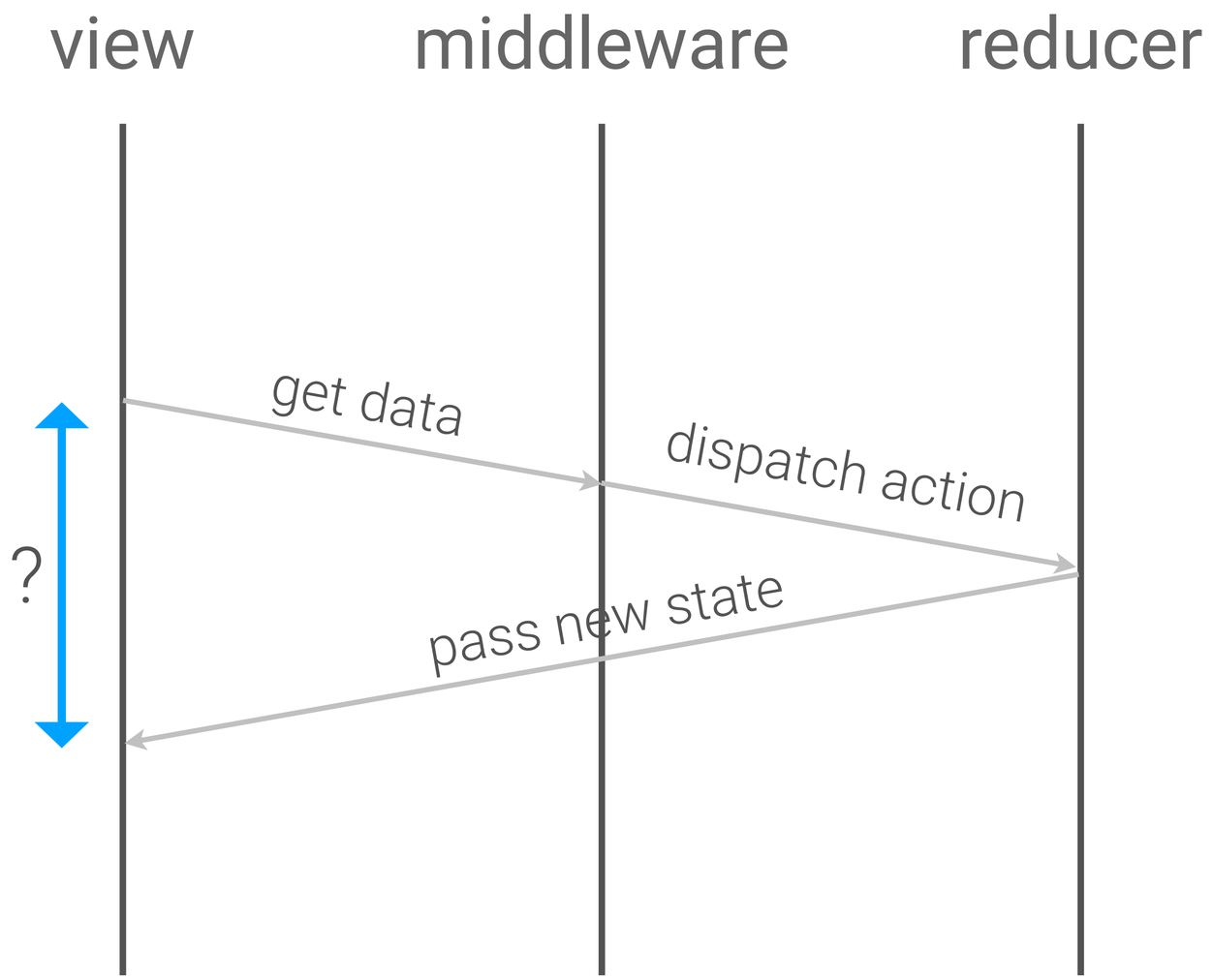
When **not** to use Redux?

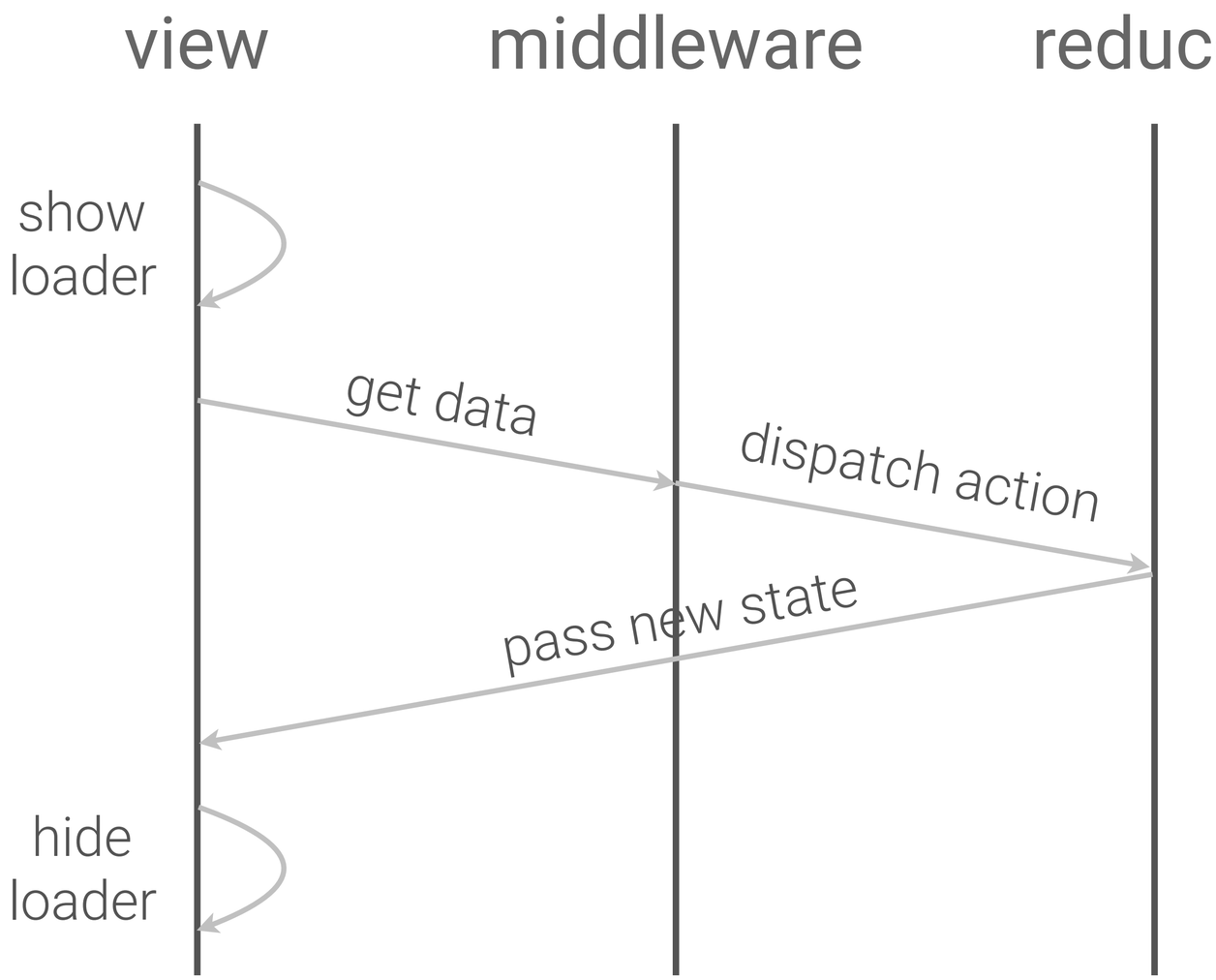
app has **no state**
state is **local**, not global


view
middleware
reducer


view
middleware
reducer
get data
dispatch action


view
middleware
reducer
get data
dispatch action
pass new state


view
middleware
reducer
get data
dispatch action
pass new state
?


view
middleware
reducer
show loader
get data
dispatch action
pass new state
hide loader

When **not** to use Redux?

app has **no state**
state is **local**, not global
other **architecture** suits you better

Want more?

Flutter + Redux - How to make shopping list app?
https://hackernoon.com/flutter-redux-how-to-make-shopping-list-app-1cd315e79b65

Brian Egan: Keep It Simple, State
https://www.youtube.com/watch?v=zKXz3pUkw9A

Thanks!

Questions?