BUILDING

MAINTAINABLE & SCALABLE APPS WITH

ANGULAR + REDUX

USE THE RIGHT TOOL FOR THE JOB

USE THE RIGHT TOOL FOR THE JOB

USE THE RIGHT TOOL FOR THE JOB

USE THE RIGHT TOOL FOR THE JOB

USE THE RIGHT TOOL FOR THE JOB

REDUX MIGHT BE THE RIGHT TOOL IF…

REDUX MIGHT BE THE RIGHT TOOL IF…

- User workflows are complex

REDUX MIGHT BE THE RIGHT TOOL IF…

- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)

REDUX MIGHT BE THE RIGHT TOOL IF…

- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate

REDUX MIGHT BE THE RIGHT TOOL IF…

- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate
- You’re using web sockets or Server Sent Events

REDUX MIGHT BE THE RIGHT TOOL IF...

- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate
- You're using web sockets or Server Sent Events
- You're loading data from multiple endpoints to build a single view

REDUX MIGHT BE THE RIGHT TOOL IF…

- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate
- You're using web sockets or Server Sent Events
- You're loading data from multiple endpoints to build a single view

Otherwise…

REDUX MIGHT BE THE RIGHT TOOL IF…

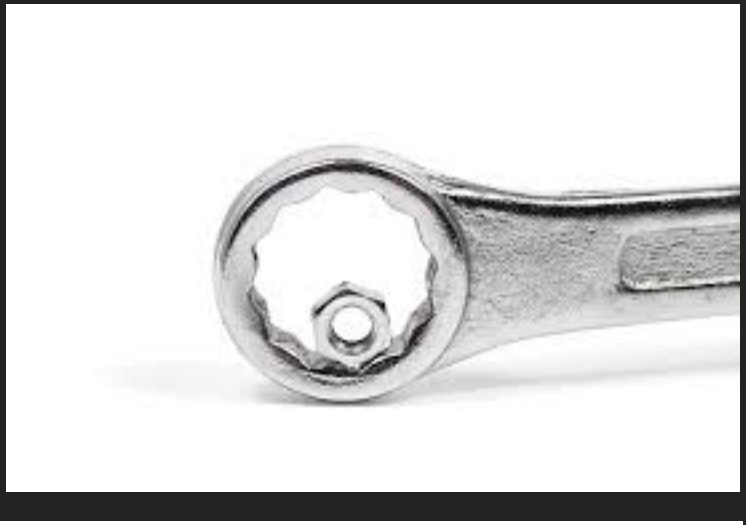
- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate
- You're using web sockets or Server Sent Events
- You're loading data from multiple endpoints to build a single view

Otherwise…

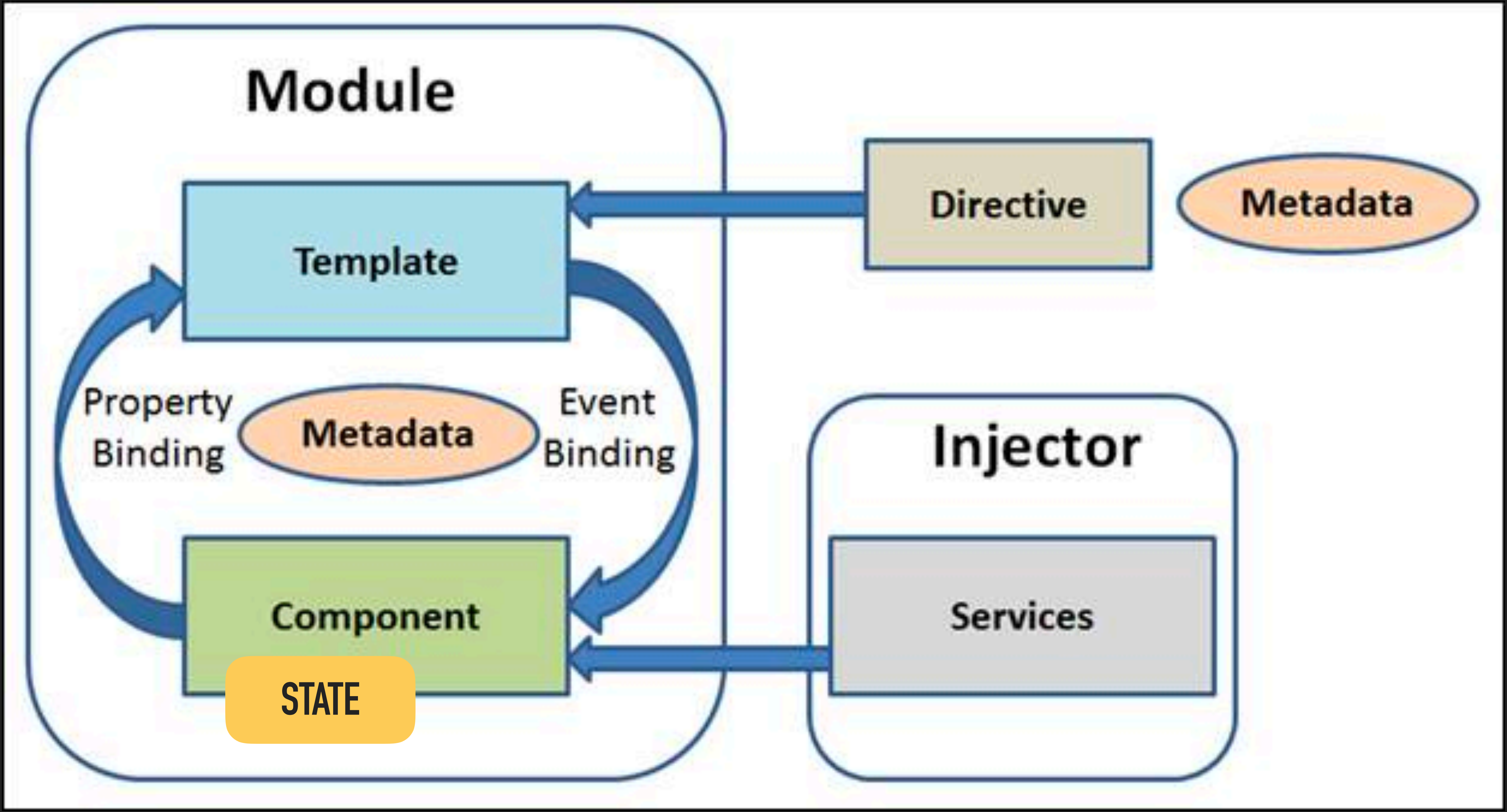
REDUX MIGHT BE THE RIGHT TOOL IF…

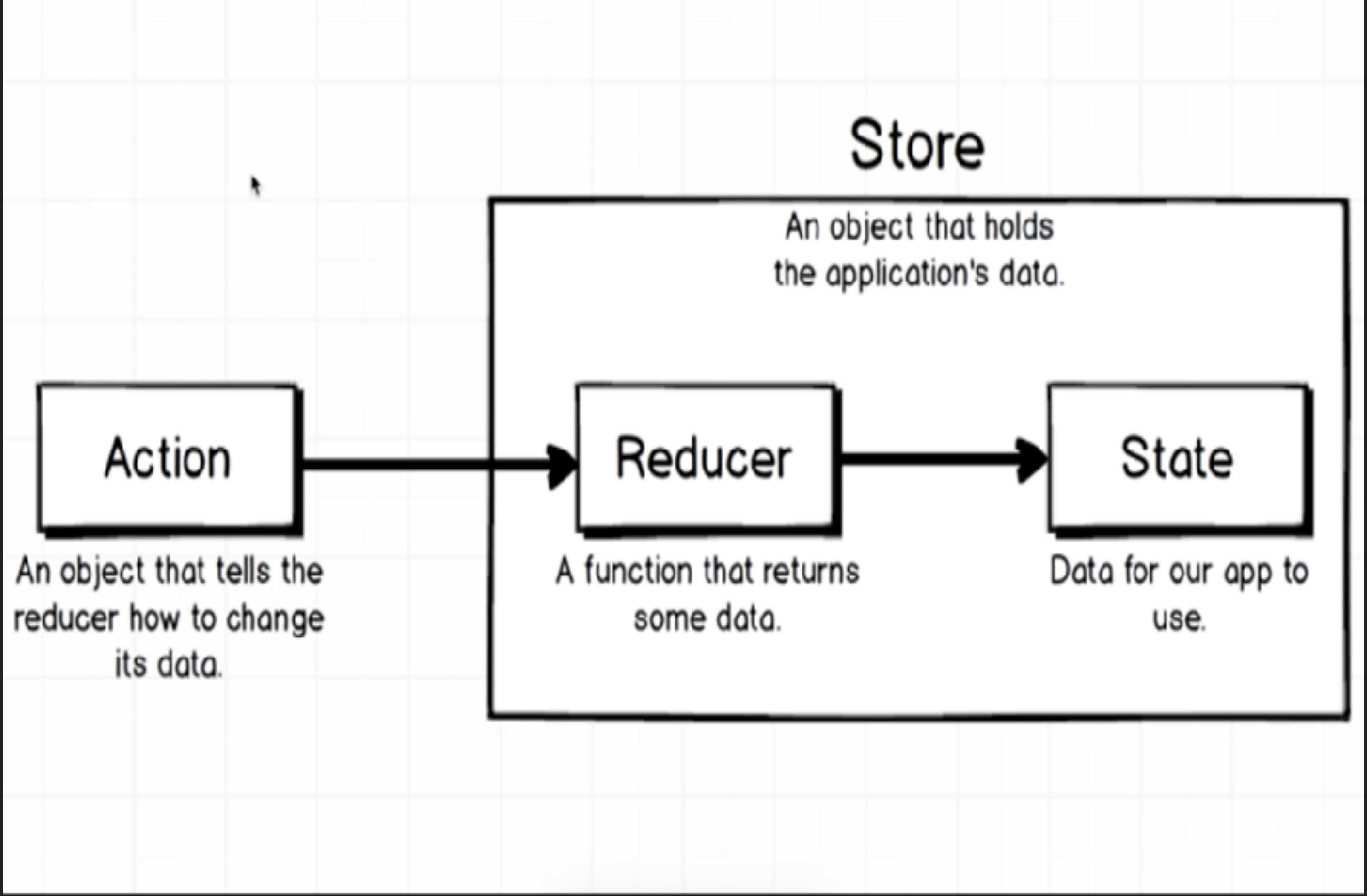
- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate
- You're using web sockets or Server Sent Events
- You're loading data from multiple endpoints to build a single view

Otherwise…



BASIC ANGULAR (V2+) ARCHITECTURE





Store

An object that holds
the application's data.

Action

An object that tells the
reducer how to change
its data.

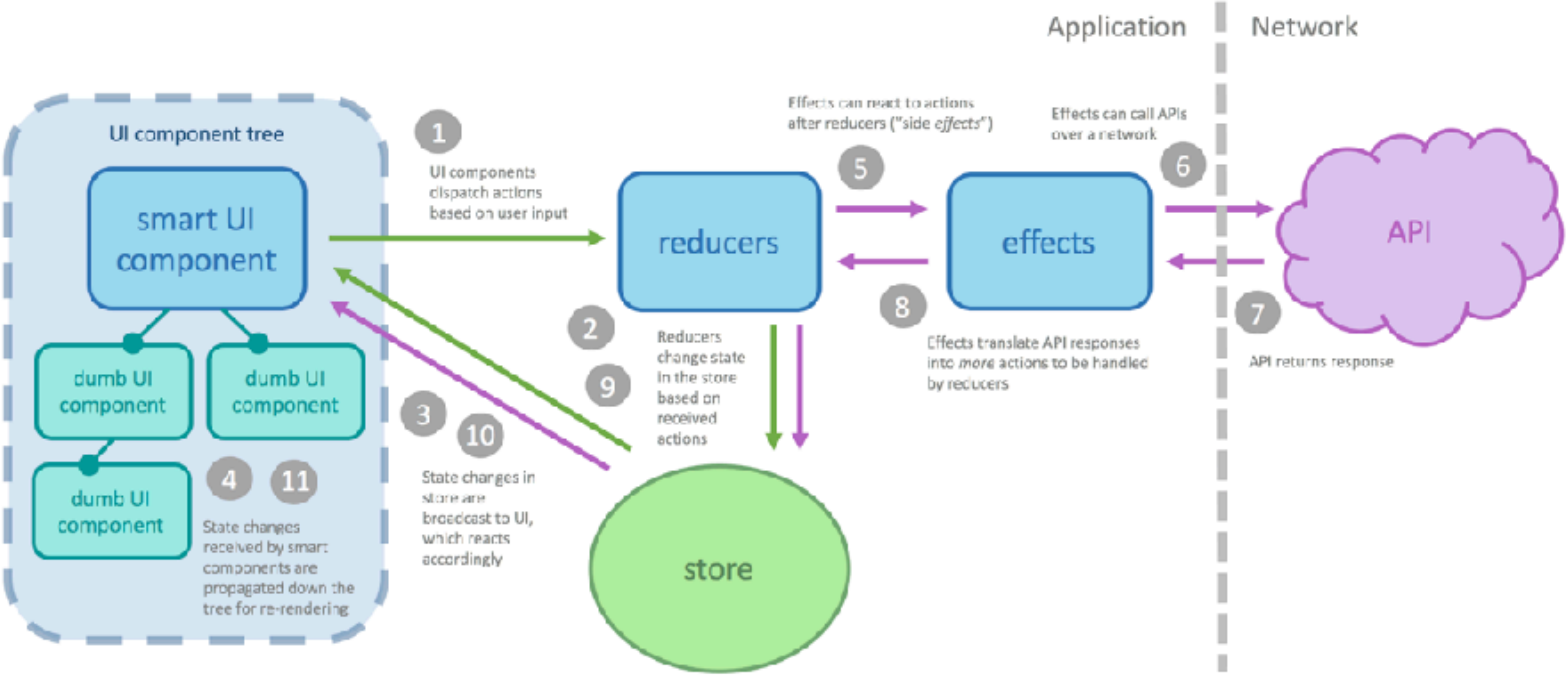
Reducer

A function that returns
some data.

State

Data for our app to
use.

Architecture – Data Flow



UI components dispatch actions based on user input

Reducers change state in the store based on received actions

State changes in store are broadcast to UI, which reacts accordingly

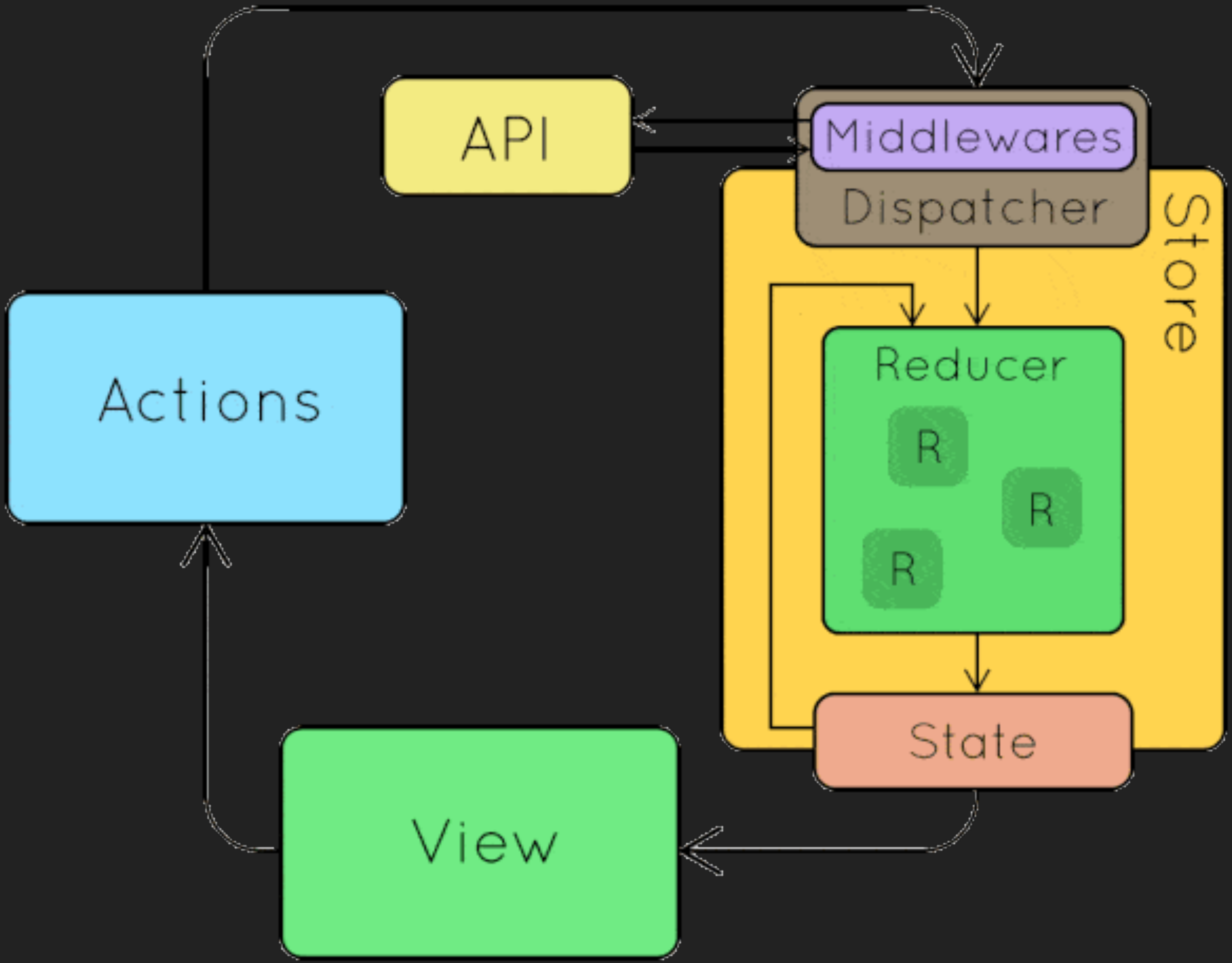
State changes received by smart components are propagated down the tree for re-rendering

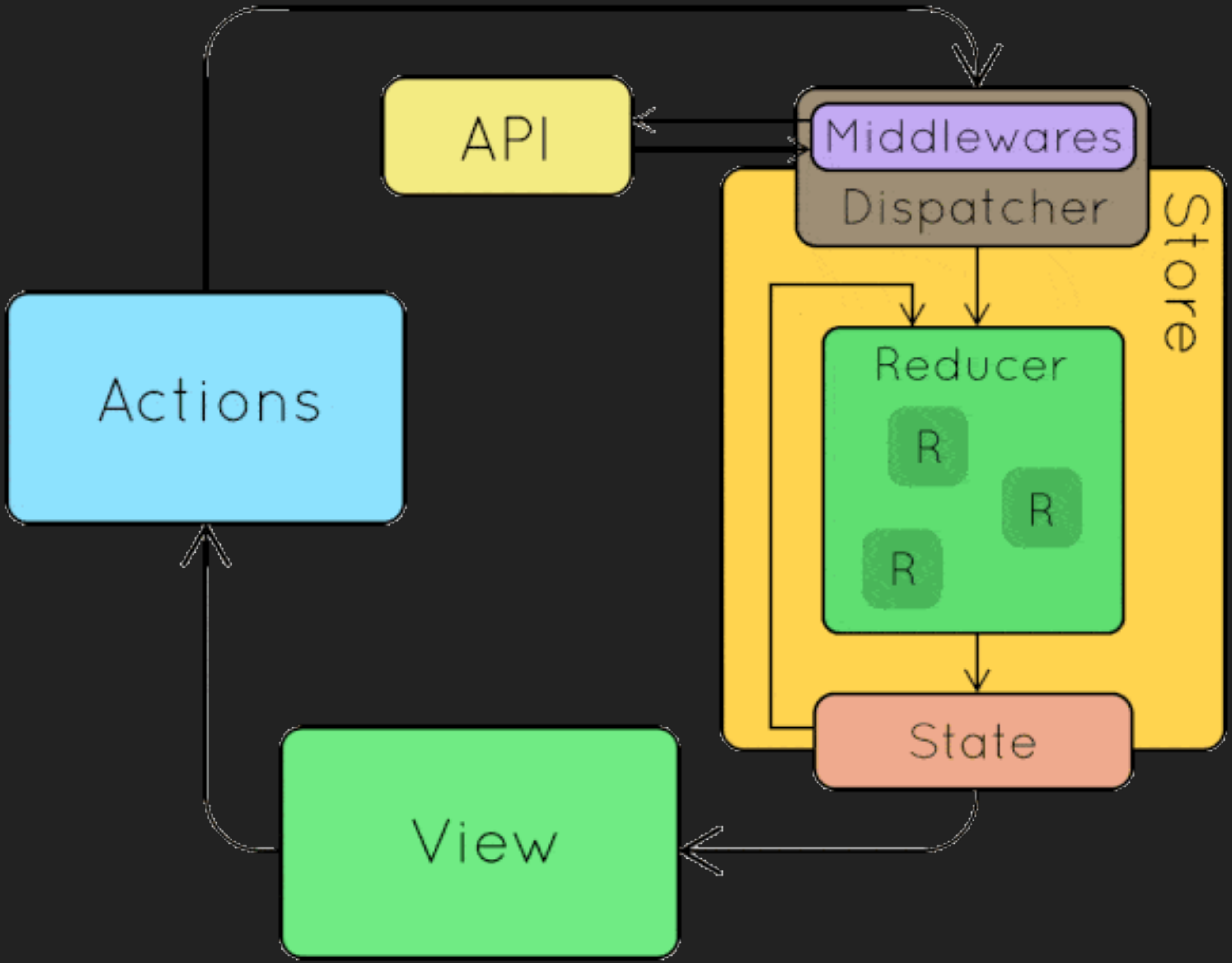
Effects can react to actions after reducers ("side *effects*")

Effects can call APIs over a network

API returns response

Effects translate API responses into *more* actions to be handled by reducers


API
Middlewares
Dispatcher
Store
Actions
Reducer
R
R
R
State
View


API
Middlewares
Dispatcher
Store
Actions
Reducer
R
R
R
State
View

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

PROVIDE THESE BENEFITS:

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store

PROVIDE THESE BENEFITS:

BENEFITS OF REDUX

PROVIDE THESE BENEFITS:

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders
- Deterministic State Reproduction

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders
- Deterministic State Reproduction
- State updates are transactional

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders
- Deterministic State Reproduction
- State updates are transactional
- Testing is easier

BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders
- Deterministic State Reproduction
- State updates are transactional
- Testing is easier
- More performant - reducing Angular change detection, onPush: assuming all inputs are immutable

Additional resources:

https://github.com/apasternack/Presentations/tree/master/AngularRedux

THE END

STAY IN TOUCH

| Adam Pasternack

| Twitter: @AJPasternack

| adam.pasternack@gmail.com