React, Redux, Trivia

Dean Radcliffe (@deaniusaur)

work w/ Test Double (@testdouble)

~~React, Redux, Trivia~~

Redux, ES6

almost a Trivia App

- https://github.com/deanius/react-trivia

Where I got stuck

- Reducer/combineReducers
- ES6 **`export default`**, **`babel-node`**
- ES6 module load order
- Redux DevTools
- no time for Animation :(

What was Easy!

- Redux-act / Flux Standard Actions
- WebSocket action replication
- App extending

Trivia

JavaScript took a different amount time to develop compared to most languages— it took:

- 10 hours
- 10 days
- 10 weeks
- 10 months

Trivia

JavaScript took a different amount time to develop compared to most languages— it took:

- 10 hours
- **10 days**
- 10 weeks
- 10 months

Trivia

Mountain Dew was originally slang for:

- Coffee
- RedBull
- Whiskey
- Moonshine

Trivia

Mountain Dew was originally slang for:

- Coffee
- RedBull
- Whiskey
- **Moonshine**

Ozarks
Mountain Dew

Mtn Dew
DEWSHINE
Made with Real Sugar
Clear Citrus Flavored Dew
It'll Tickle Yore Innards!
NON-ALCOHOLIC

Reduction

- From Many, One
- *E Pluribus Unum*
- Aggregation like `SUM`
- Adds a piece to the whole

Reducers (O.G.)

```
const sumReducer = (piece1, piece2) =>
    piece1 + piece2

[1,2,4].reduce(sumReducer)
> 7    // 1+2   3+4

[1,2,4].reduce(sumReducer, 35)
> 42   // 35+1   36+2   38+4
```

Reducers

Job: Produce a new state, given an existing state, and a new data item

`(existing, item) -> new`

Use Cases:

- build up an array of all items
- a total,
- only keep most recent item

Reducers (Redux)

Must Provide initial piece if not given

Implements the action, using its payload, against the Store

```
const sumReducerRedux = (piece1, piece2) =>
    piece1 ? piece1 + piece2 : 0
```

Different reducers can 'own' different parts of the state tree

BUT, they must be set up to only listen to certain actions

Trivia

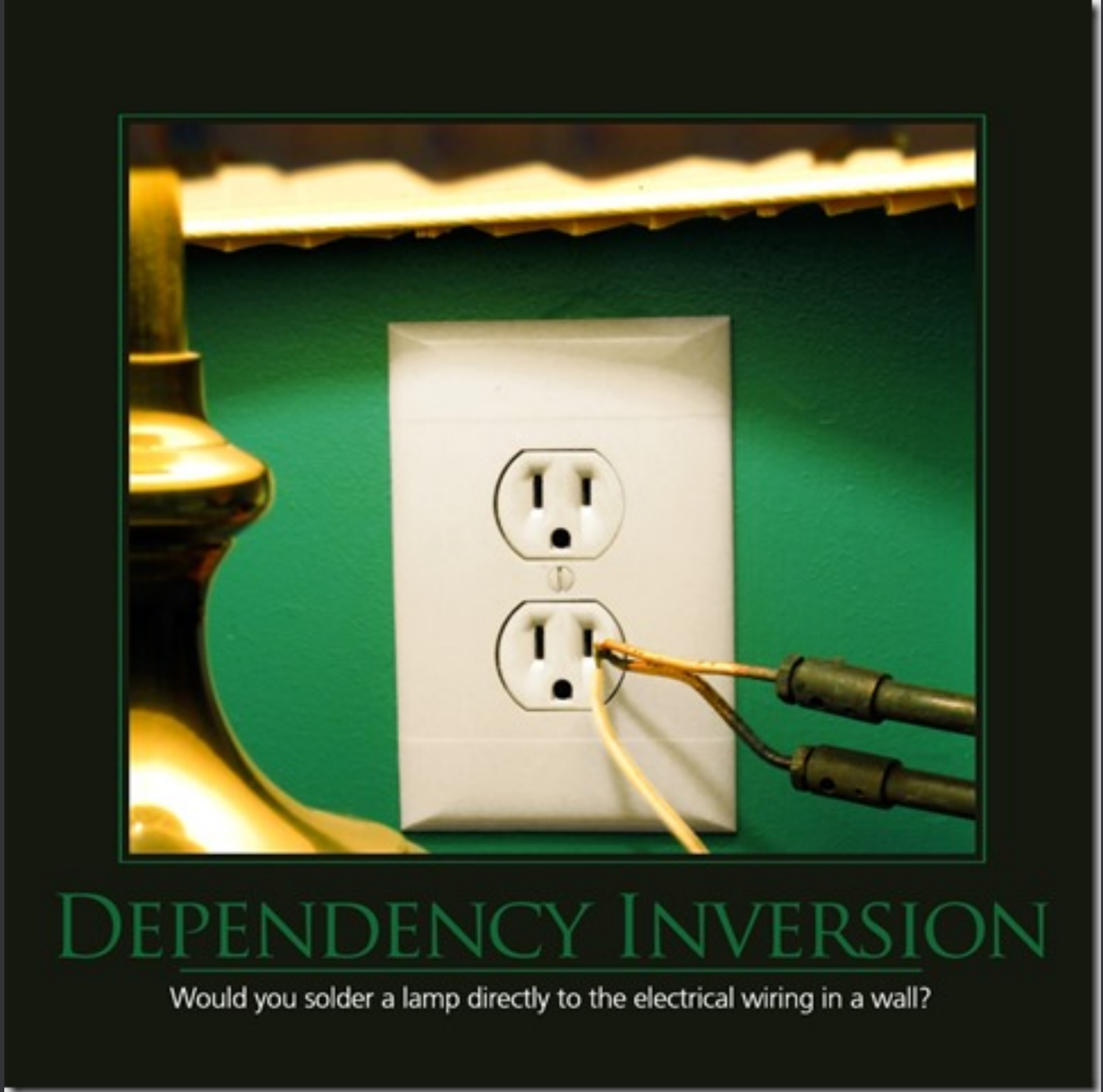
The Dependency Inversion Principle applies to Redux because

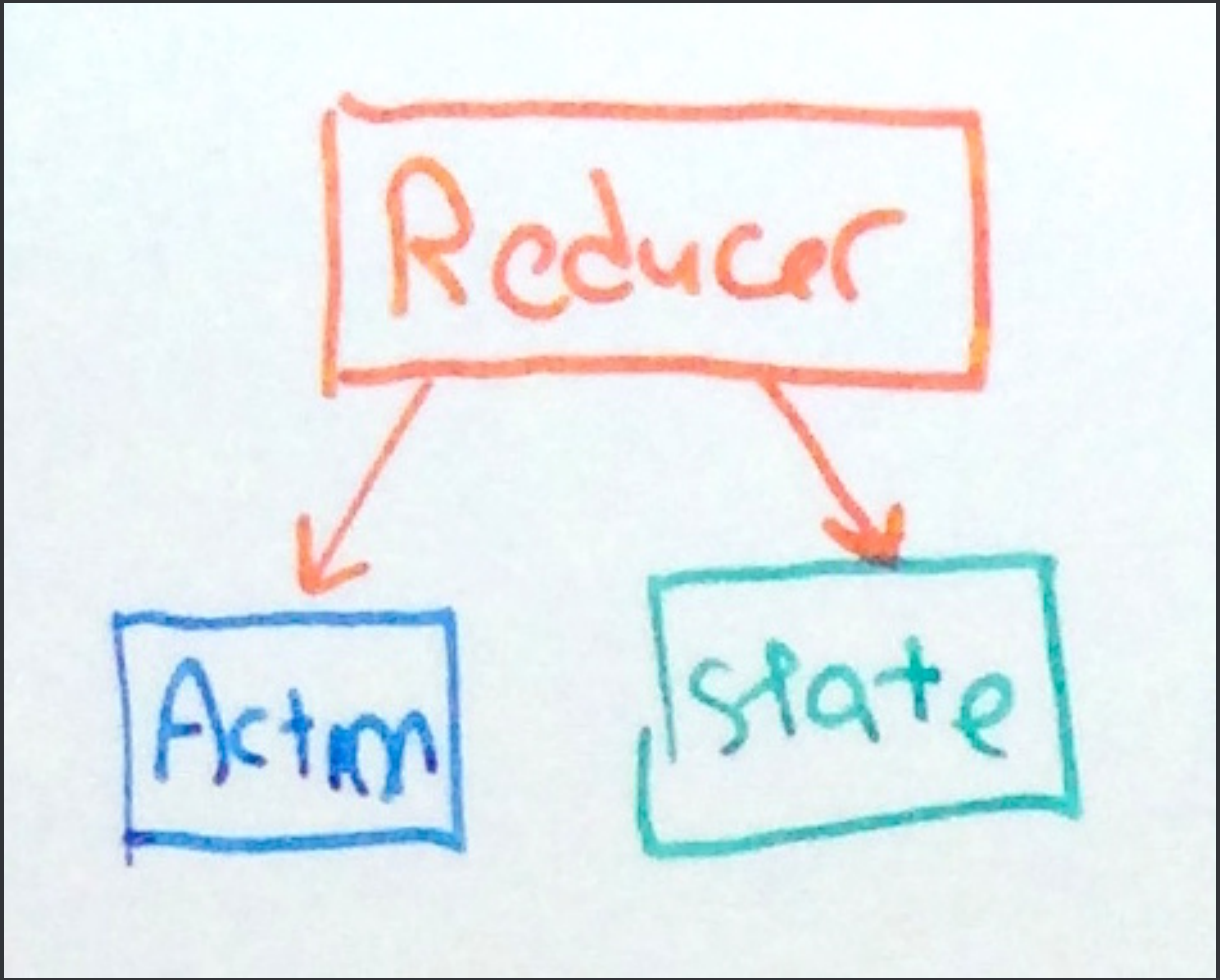
- Redux depends on React
- The Store and Reducers know about Actions
- React depends on Redux
- Huh?

Trivia

The Dependency Inversion Principle applies to Redux because

- Redux depends on React
- **The Store and Reducers know about Actions**
- React depends on Redux
- Huh?


DEPENDENCY INVERSION
Would you solder a lamp directly to the electrical wiring in a wall?


Reducer
Action
State

$$\begin{tikzcd}
 & A \arrow[d] & A' \arrow[d] & A'' \arrow[d] & \\
I \arrow[r] & S \arrow[r] & S \arrow[r] & S \arrow[r, dotted] & {}
\end{tikzcd}$$

Yuck! (the 'official' way)

```
function todoApp(state = initialState, action) {
  switch (action.type) {
    case SET_VISIBILITY_FILTER:
      return Object.assign({}, state, {
        visibilityFilter: action.filter
      })
    default:
      return state
  }
}
```

Option: `redux-act`

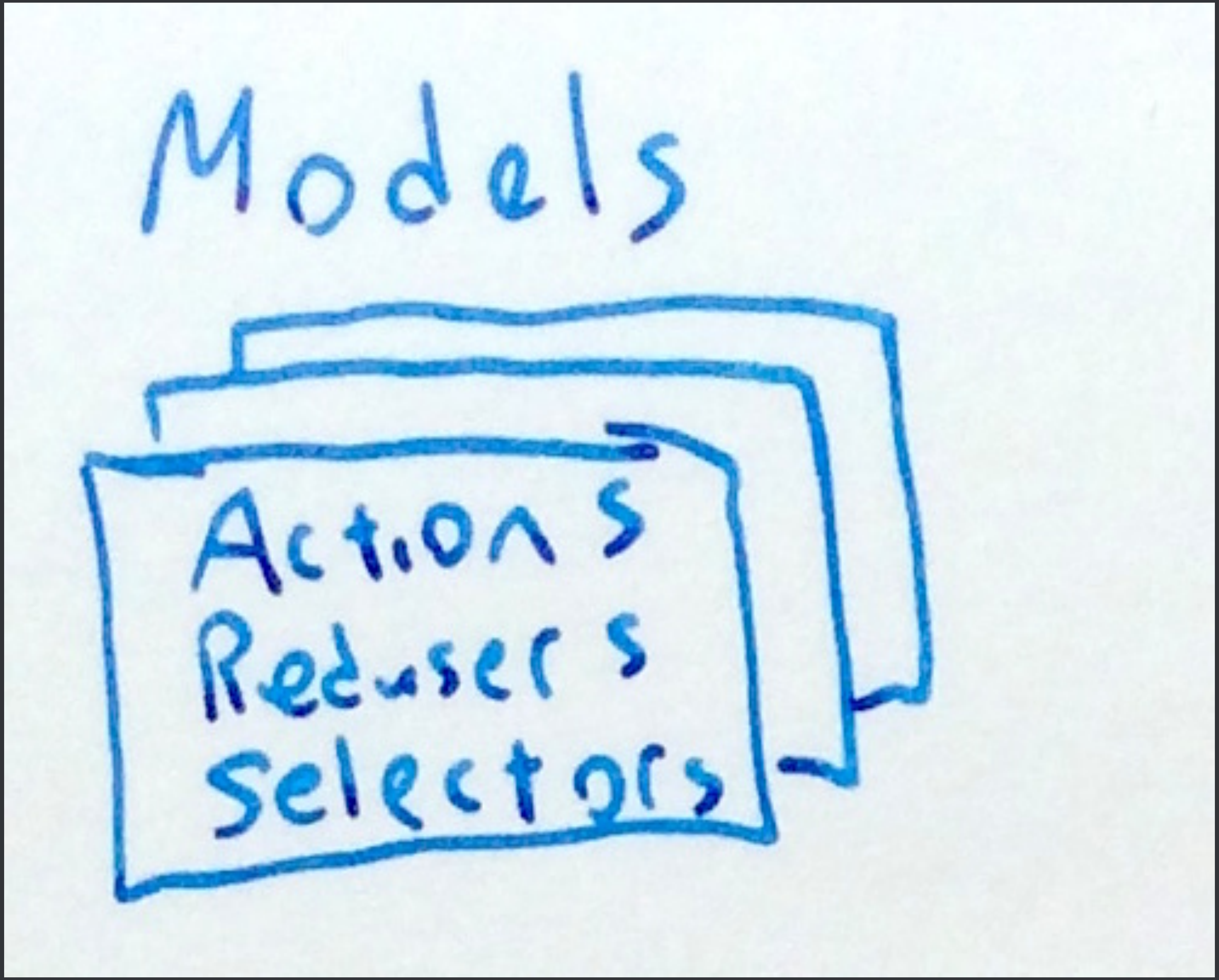
```
joinPlayer(player) =>
  {
    type: 'JOIN_PLAYER'
    payload: player
  }
```

ActionCreator / Action

FSA: Flux
Standard
Action

with 'redux-act'

```
JoinPlayer = createAction
        ('JOIN_PLAYER')
```


Models
Actions
Redusers
Selectors

Round

```
{
  question : {}
  responses : []
}  // initial state
```

Actions Actions

```
{
  answerQuestion (choice)
  advanceQuestion ( )
}
```

Selectors

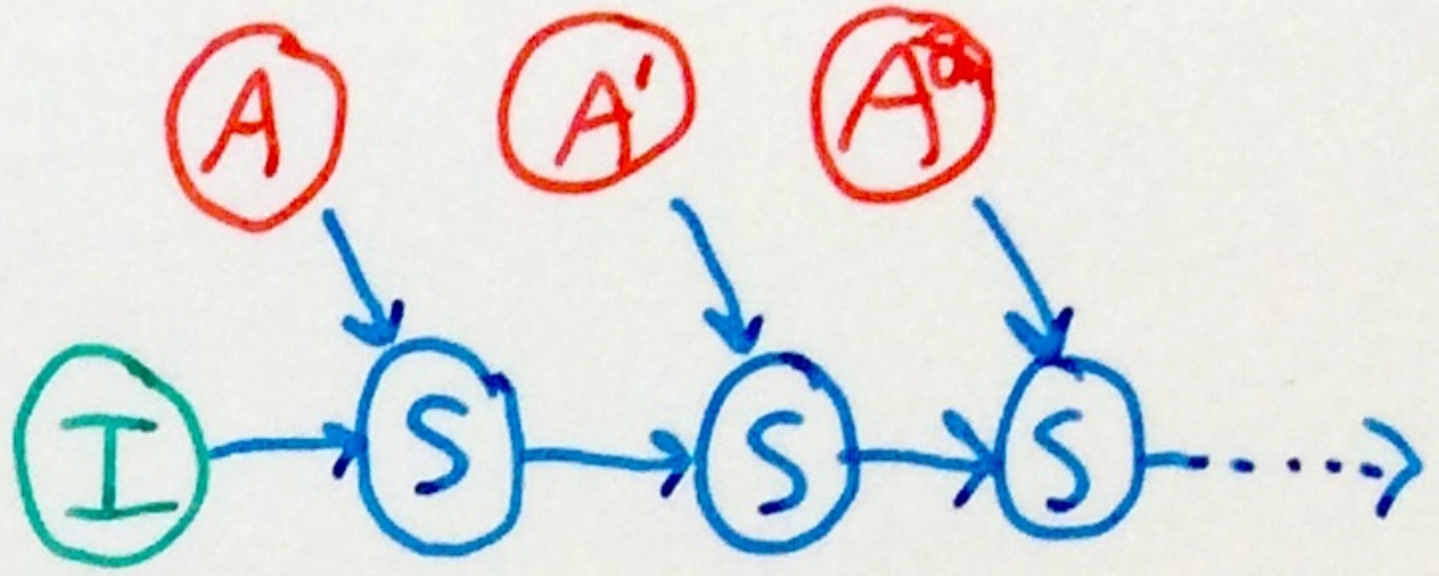
```
{
  questionIsAnswered
  answerState  //pending
                 confirmed
                 beaten
                 correct
                 incorrect
```

```
export let advanceQuestion = createAction('ADVANCE_QUESTION')

export let initialRound = {question: null, responses: null}

export let Actions = {
  advanceQuestion
}

export let Reducer = createReducer({
  [advanceQuestion]: (round, _) => ({...round, question: Question.nextQuestion(round.question)})
}, initialRound);
```



```
let reducer = createReducer({
    a: (state, payload) => State
   a1:                 "
}, initialState)
```

Trivia

Which company first introduced XHR, the foundation of AJAX?

- Google
- Netscape
- Microsoft
- Apple

Trivia

Which company first introduced XHR, the foundation of AJAX?

- Google
- Netscape
- **Microsoft**
- Apple

Realtime

- Server has its own Redux store
- Applies actions it recieves via WS

Realtime Client Middleware

On new actions, send to server

```
let middleware = store => next => (action) => {
  let sendToServer = !(action.meta && action.meta.clientOnly)
  if (sendToServer) {
    socket.emit('action', action)
  }
  return next(action)
}
```

Realtime server

```
socket.on('action', (e) => {
  store.dispatch(e)
})

store.subscribe(() => {
  var state = store.getState()
  io.emit('state', state)
})
```

Realtime on Client

On incoming state from server, merge

```
socket.on('state', Actions.setState)
```

Thank You!

Dean Radcliffe

@deaniusaur

Resources

Home of this app

http://bit.ly/react-trivia-src

Inspiration for this trivia app

http://teropa.info/blog/2015/09/10/full-stack-redux-tutorial.html

How Redux Connect works

https://gist.github.com/gaearon/