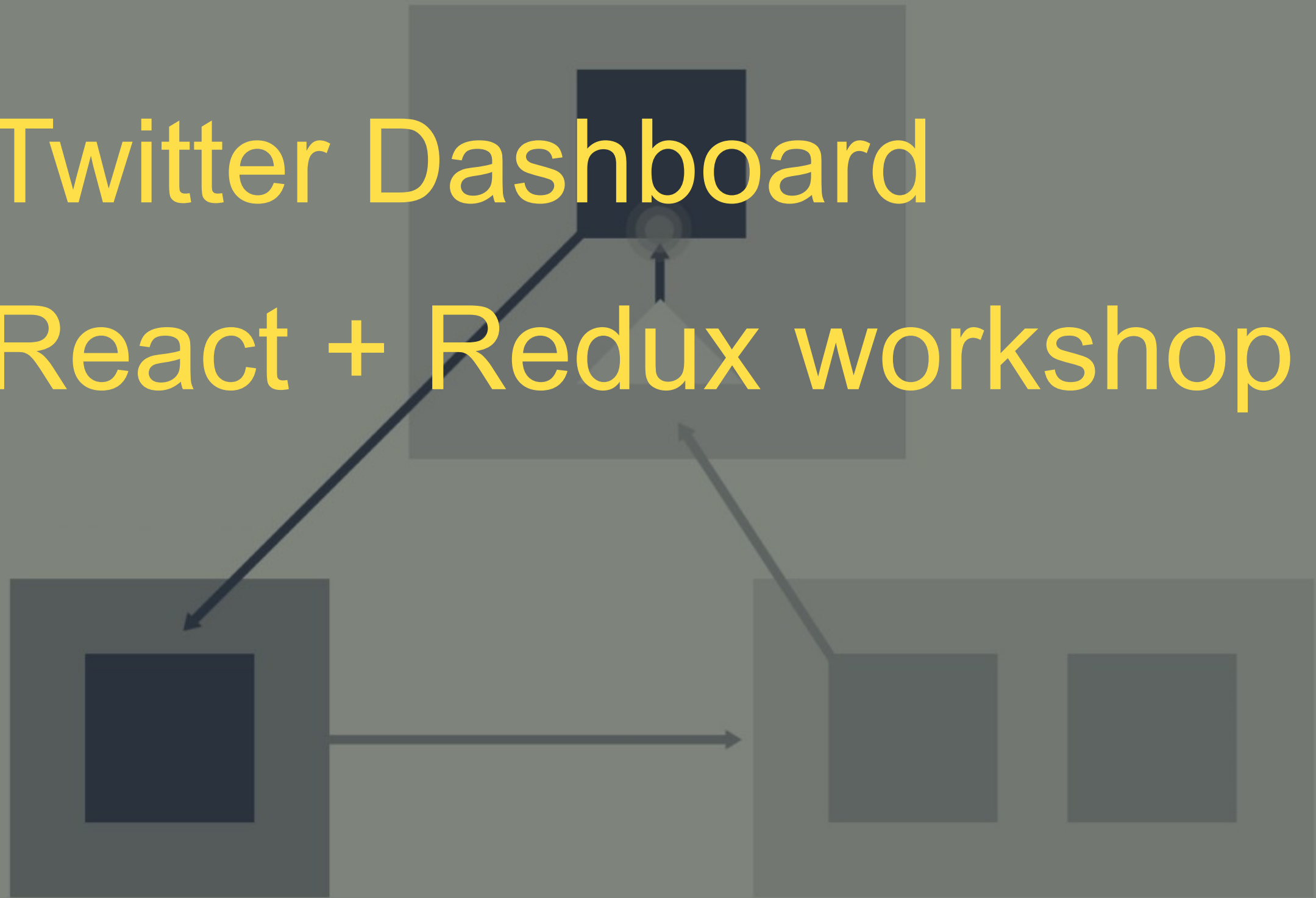


# Twitter Dashboard

## React + Redux workshop



// Guro Seternes

# Agenda


1. Presentasjon av Redux-basics
2. Dere koder Redux
3. Redux + React
2. Koding resten av dagen



ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

Live Reactx

localhost:3000



**Dan Abramov**  
@dan\_abramov


Oh no am I really writing my own Flux library

RETWEETS


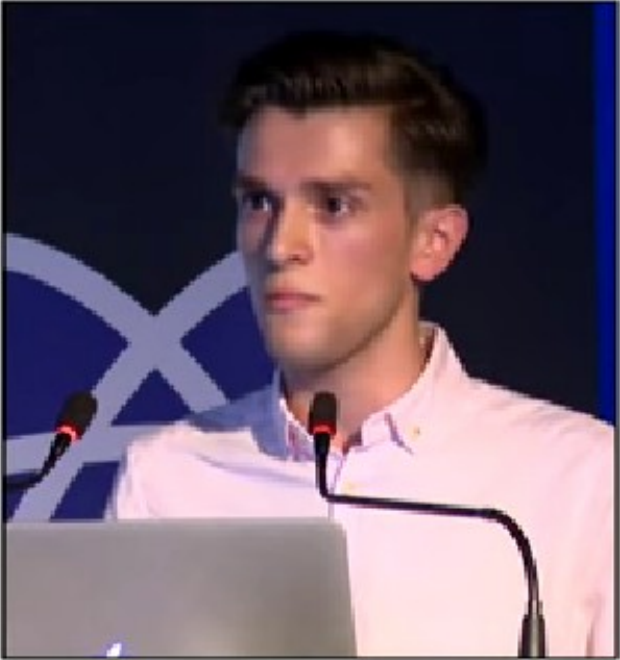
4

FAVORITES

14



10:41 PM - 29 May 2015



RANGLE.IO  
REWRITING THE WEB

17:15 / 30:40

CCSettingsFullscreen

## Dan Abramov - Live React: Hot Reloading with Time Travel at react-europe 2015



ReactEurope

 **Subscribe**

4,779

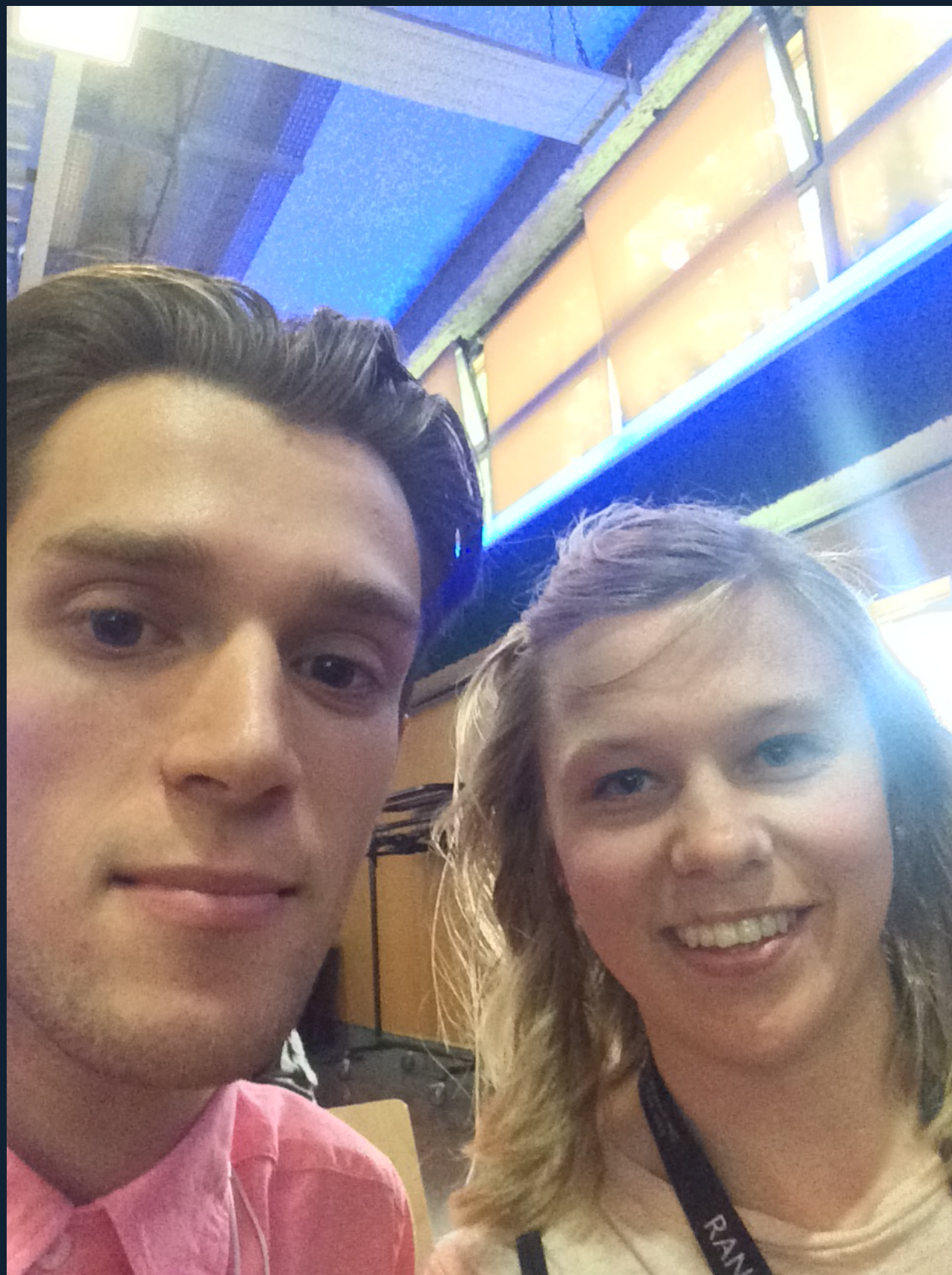
130,362

 Add to  Share  More

 2,087  4



Mannen  
bak Redux



# Kvaliteter

- » funksjonelt inspirert
- » svært testbart
- » klar fordeling av ansvar
- » egnet for store kodebaser med lang levetid



# Eksempler på bad practice

```
var numberOfTodos= 0;  
;  
function numberOfTodosIsMoreThan(val) {  
    return numberOfTodos > val;  
}  
  
;  
function addTodoItem(newItem, items) {  
    return items.push(newItem)  
}
```

# The Gist

- » The whole state of your app is stored in a single object tree.
- » The only way to change the state tree is to emit an action, an object describing what happened.
- » To specify how the actions transform the state tree, you write stateless (pure) reducers.

[redux.js.org](https://redux.js.org)



# todos

▼ *What needs to be done?*

☐ buy groceries

☐ water plants

☒ ~~prepare workshop~~

2 items left

All

Active

Completed

Clear completed

# The Gist

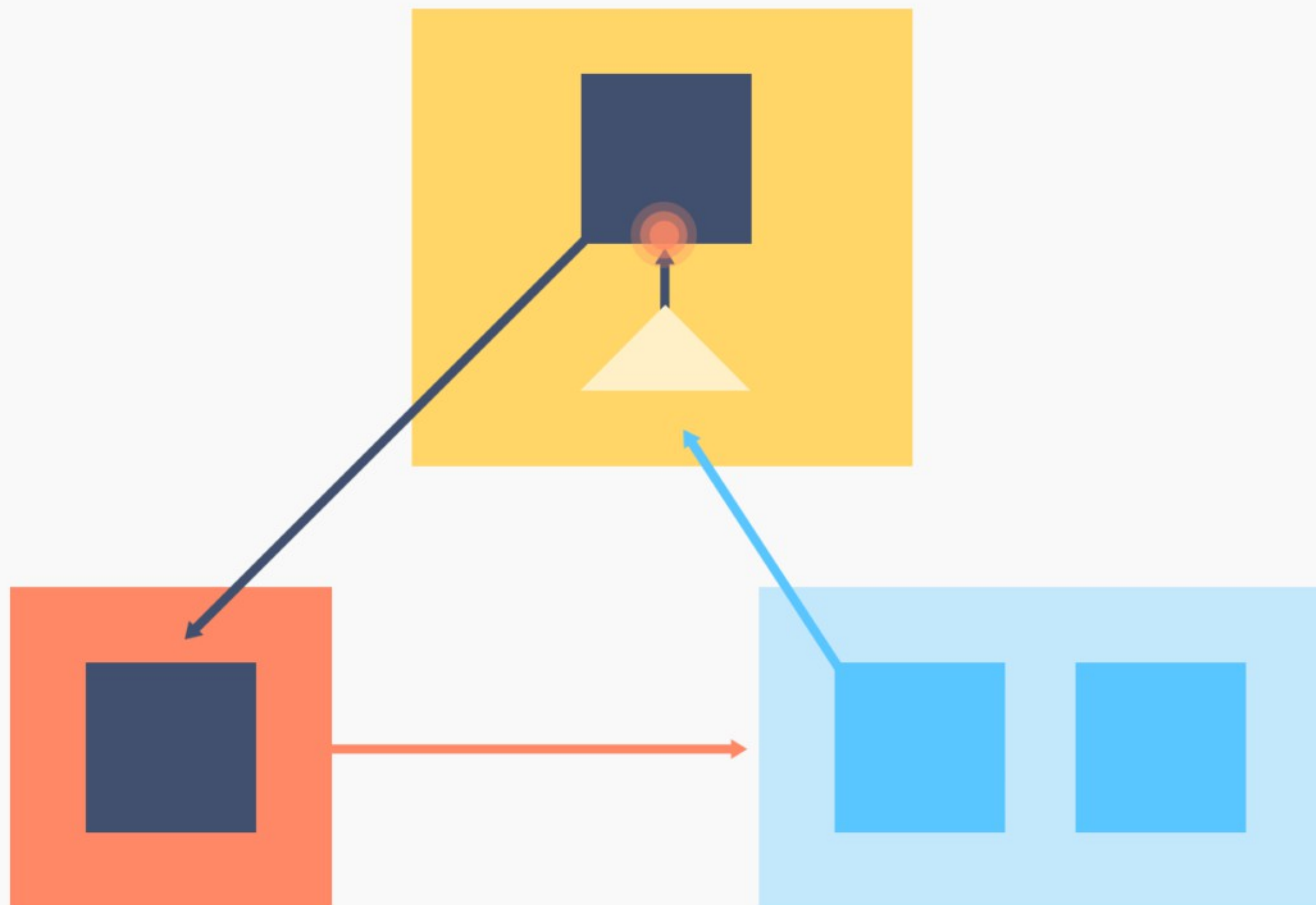
- » The whole state of your app is stored in a single object tree
- » The only way to change the state tree is to emit an action, an object describing what happened.
- » To specify how the actions transform the state tree, you write stateless (pure) reducers.

[redux.js.org](https://redux.js.org)

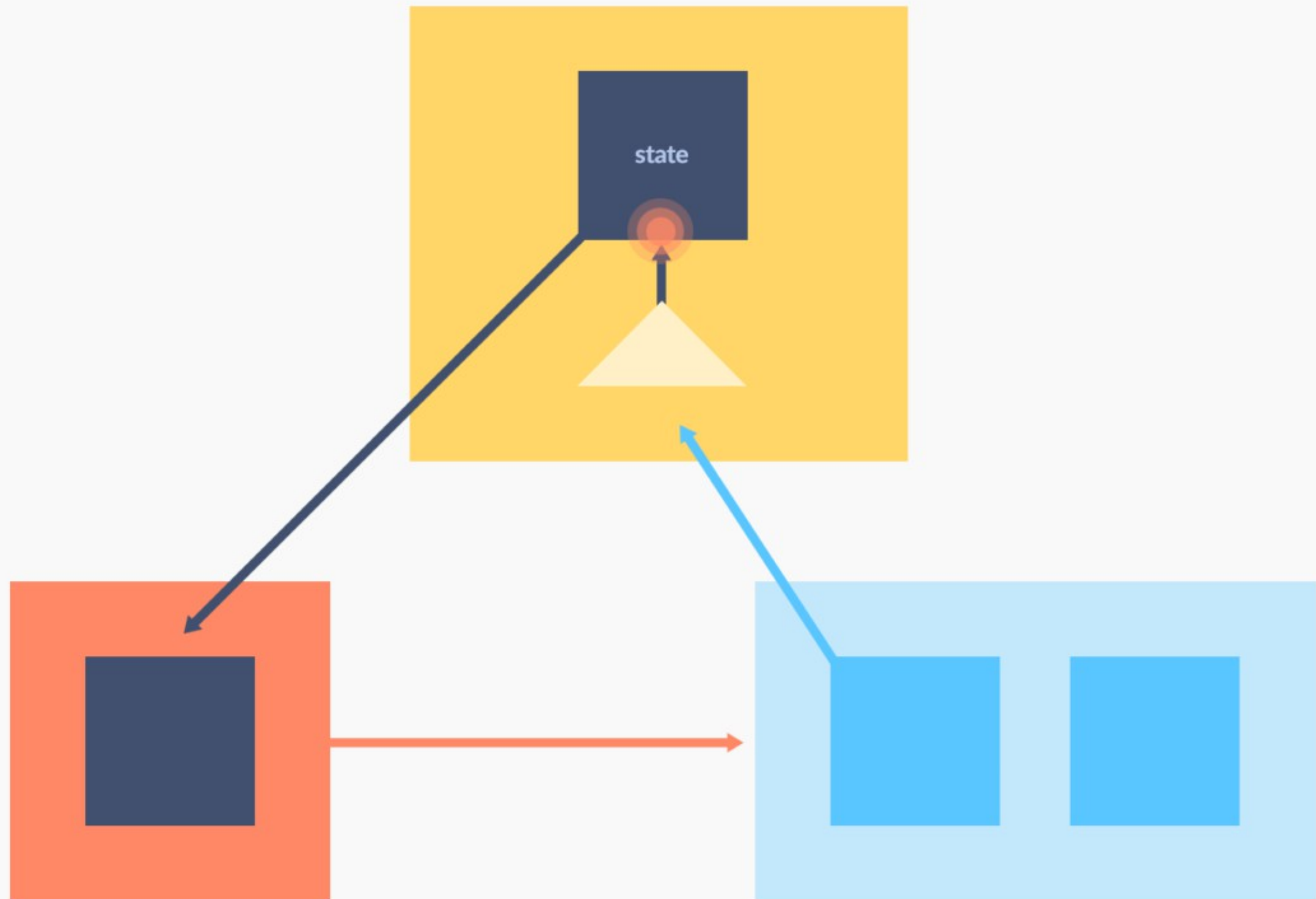
# The Gist

- » The whole state of your app is stored in a single object tree
- » The only way to change the state tree is to emit an action, an object describing what happened.
- » To specify how the actions transform the state tree, you write stateless (pure) reducers.

[redux.js.org](https://redux.js.org)



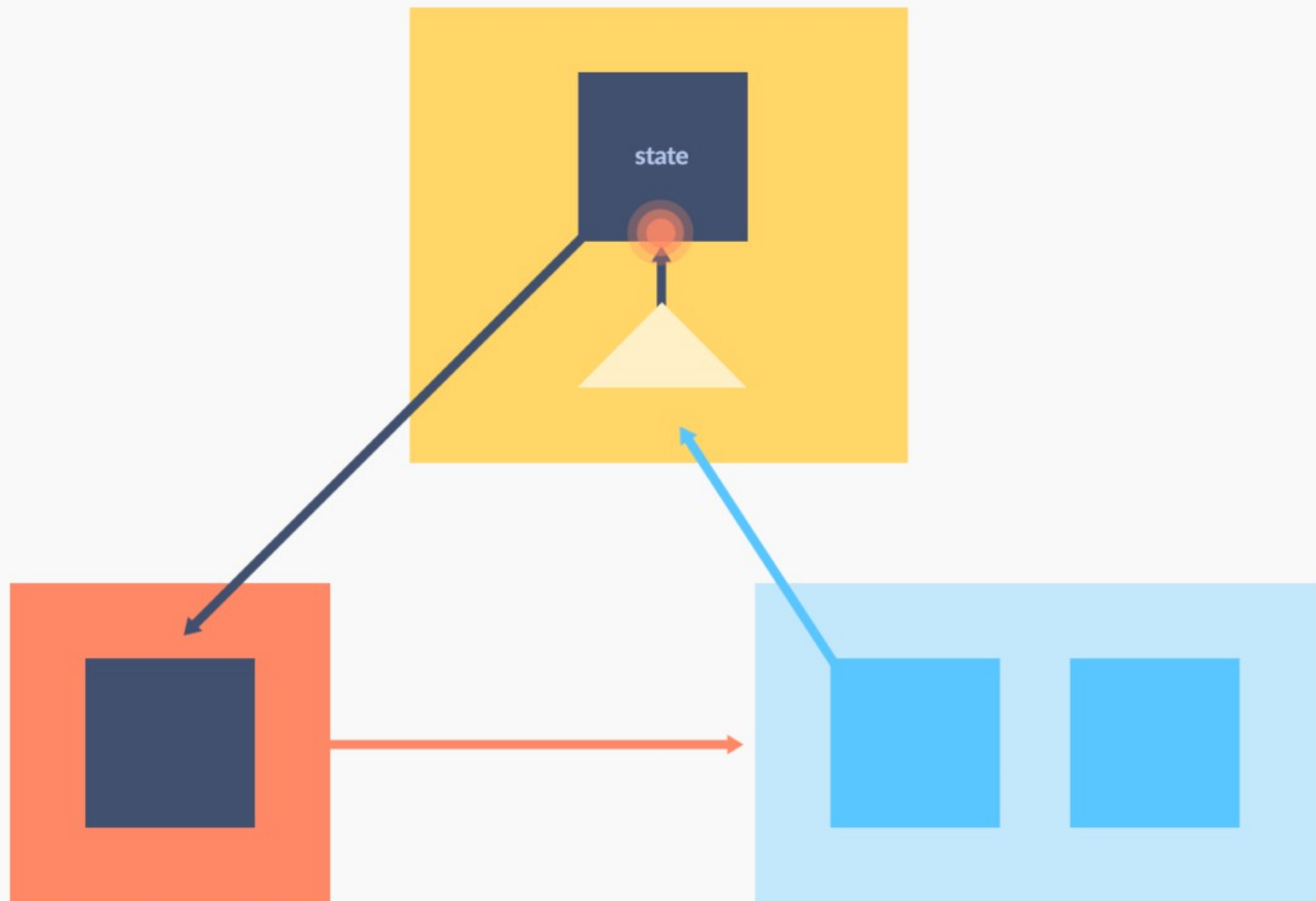
Store



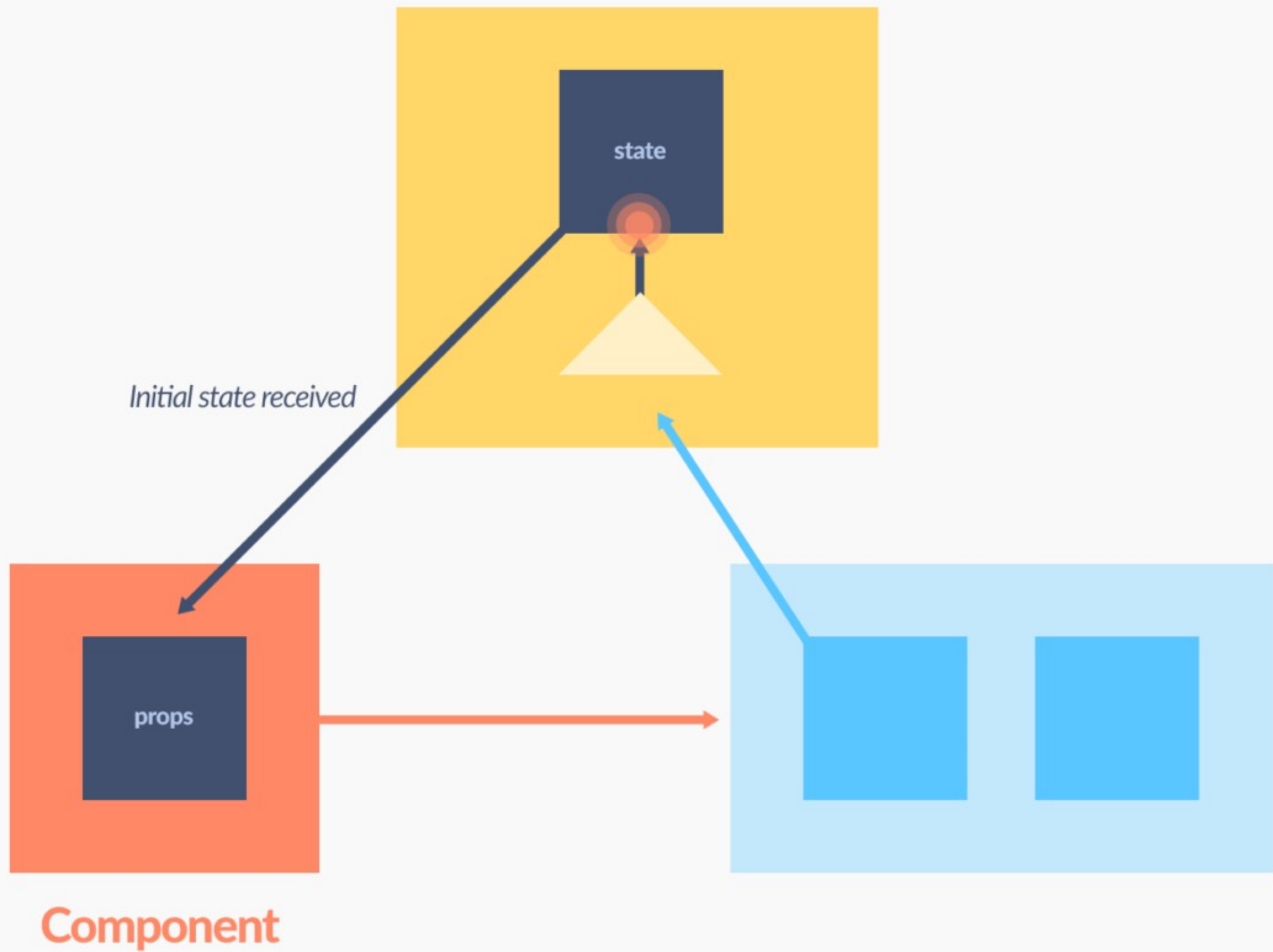
```
{  
  todos: [],  
  activeFilter: 'completed',  
}
```

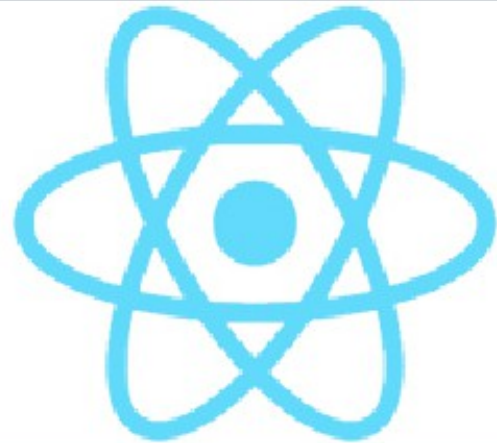


Store



Store





React

**RE-RENDER**



**ALL THE THINGS**

DATA => HTML

```
function HeadingComponent() {  
  return <div>  
    todos  
  </div>  
}
```



```
import { render } from 'react-dom';
```

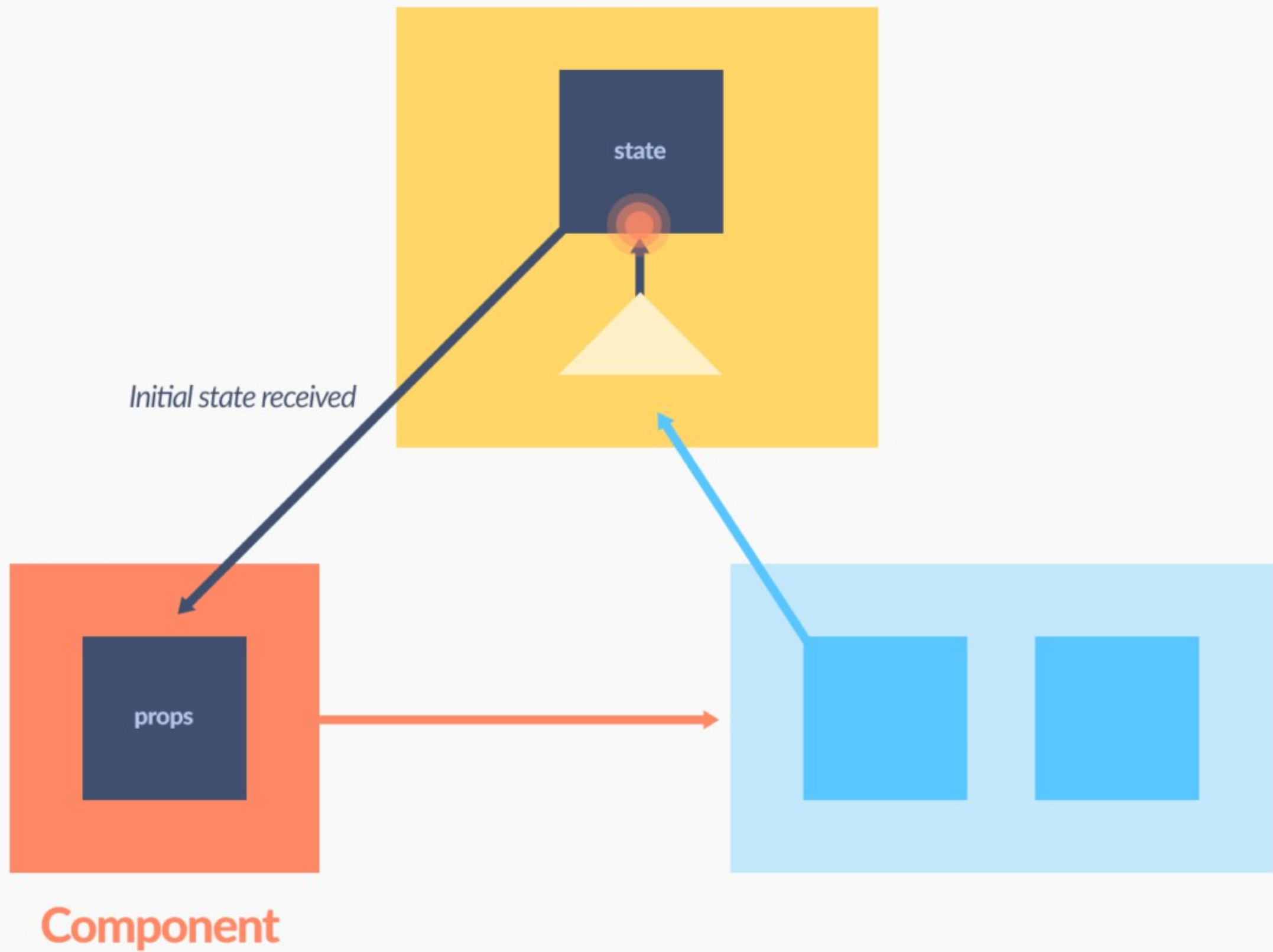
```
render(<HeadingComponent />, document.body);
```

```
function HeadingComponent(props) {  
  return <div>  
    todos for {props.name}!  
  </div>  
}
```

```
render(<HeadingComponent name="Guro" />, document.body);
```

```
function TodoAppComponent() {  
  return <div>  
    <HeadingComponent name="Guro" />  
    <TodoListComponent  
      todos={['buy groceries', 'water plants']  
    />  
  </div>;  
}
```

Store



```
{  
  todos: [],  
  activeFilter: 'completed',  
}
```

# todos

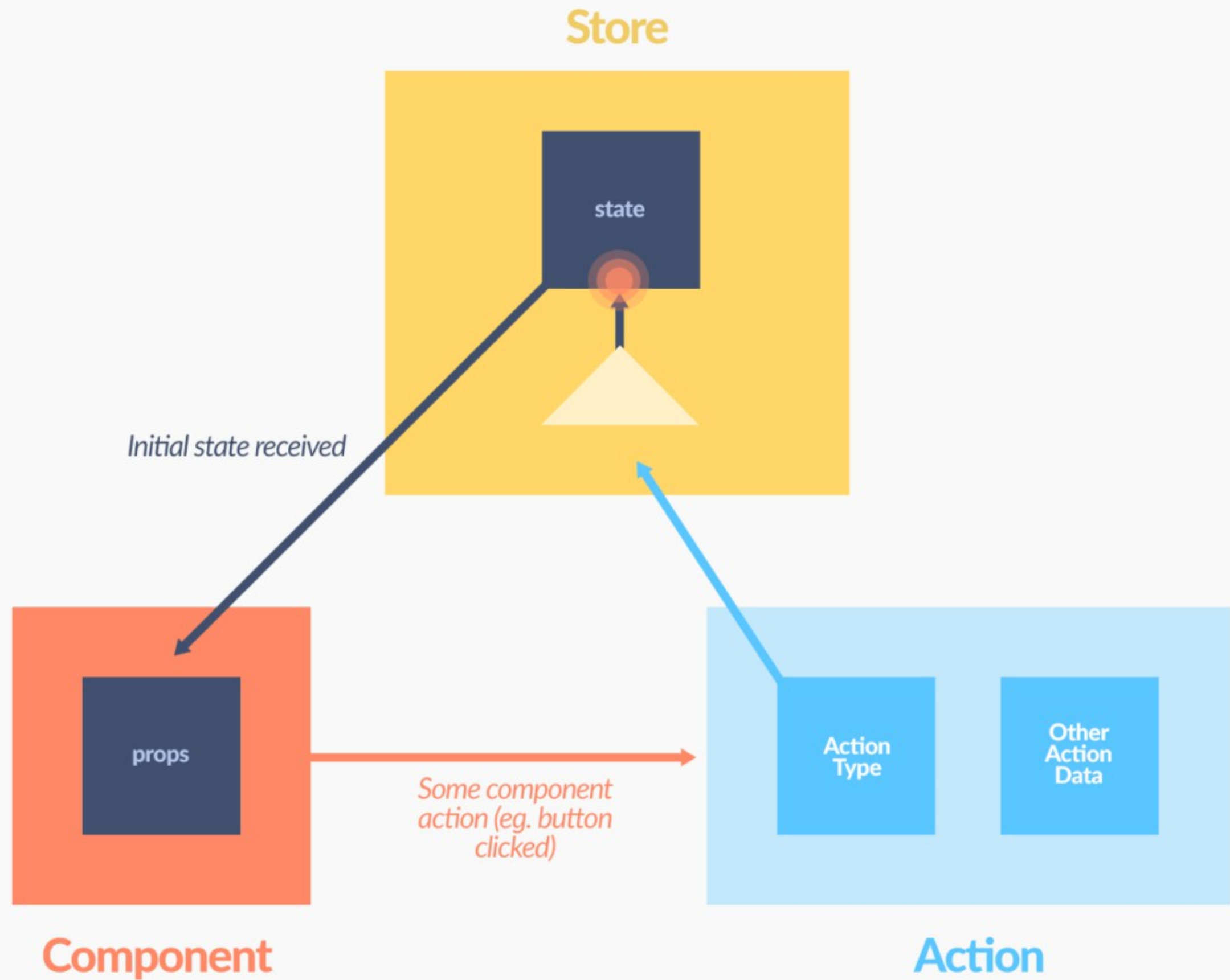
*What needs to be done?*

Double-click to edit a todo

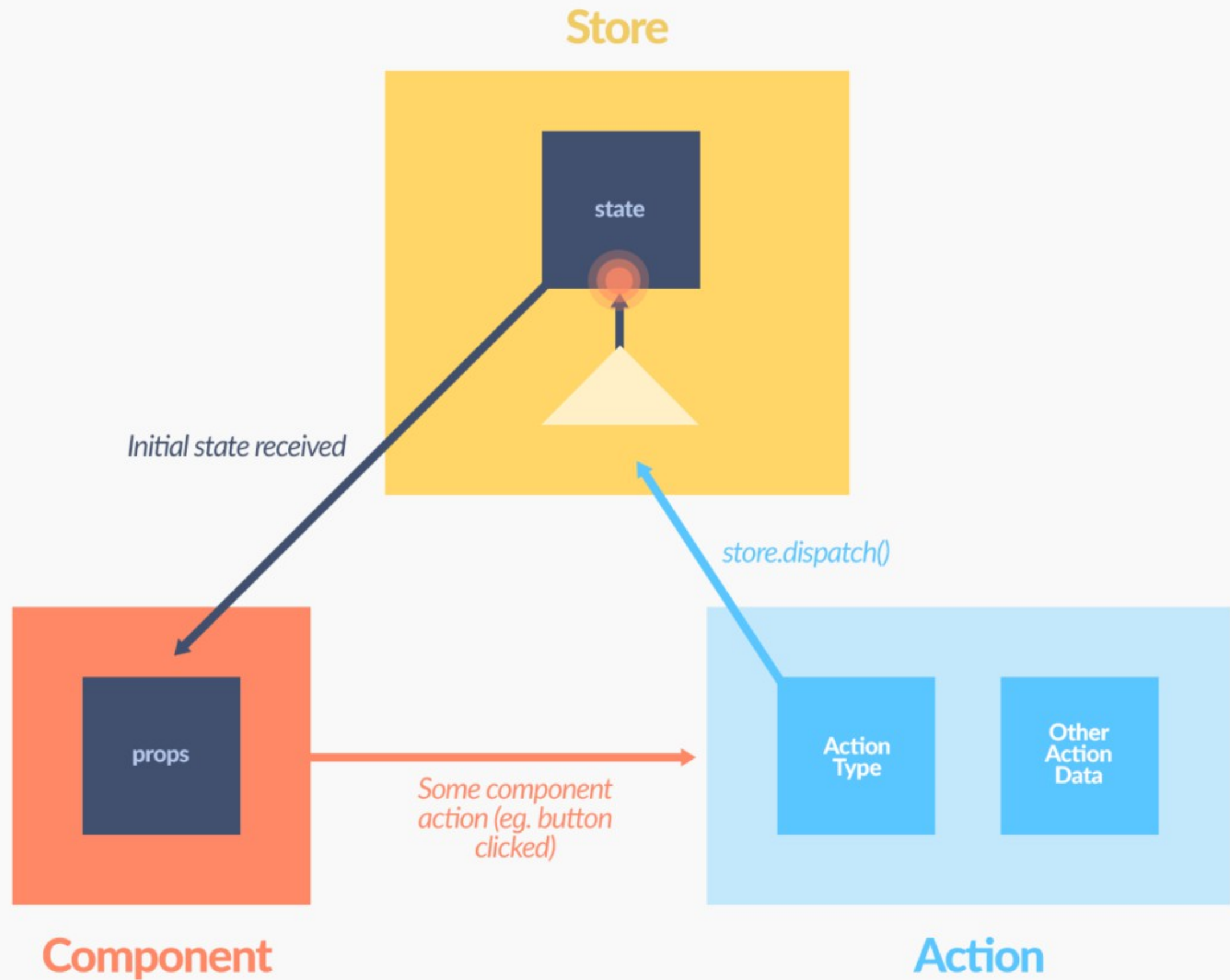
Created by [petehunt](#)

Part of [TodoMVC](#)





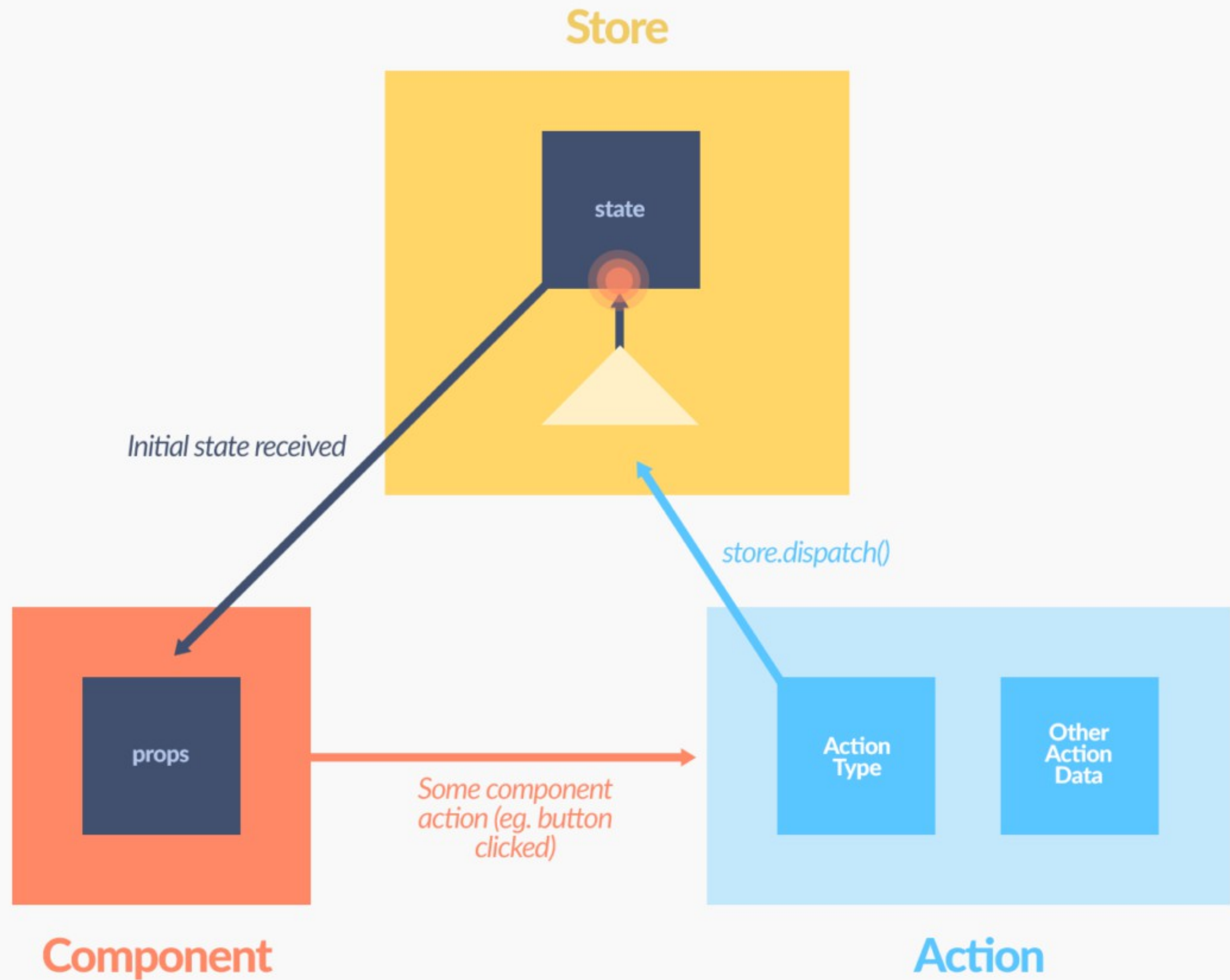
```
{  
  // action type  
  type: 'USER_ADDED_TODO',  
  // action data  
  item: 'buy groceries'  
}
```



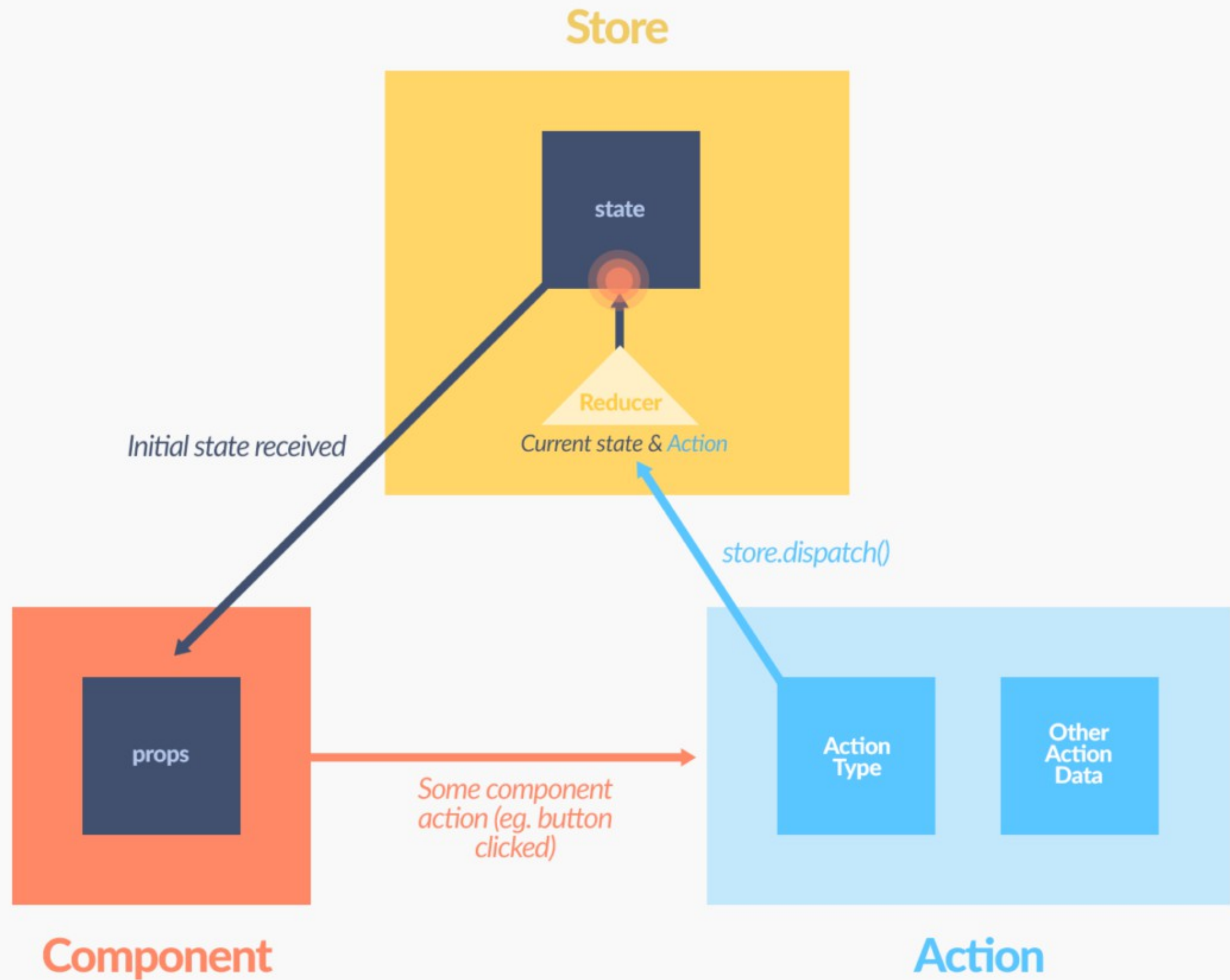
```
store.dispatch({  
  type: 'USER_ADDED_TODO',  
  item: 'buy groceries'  
})
```

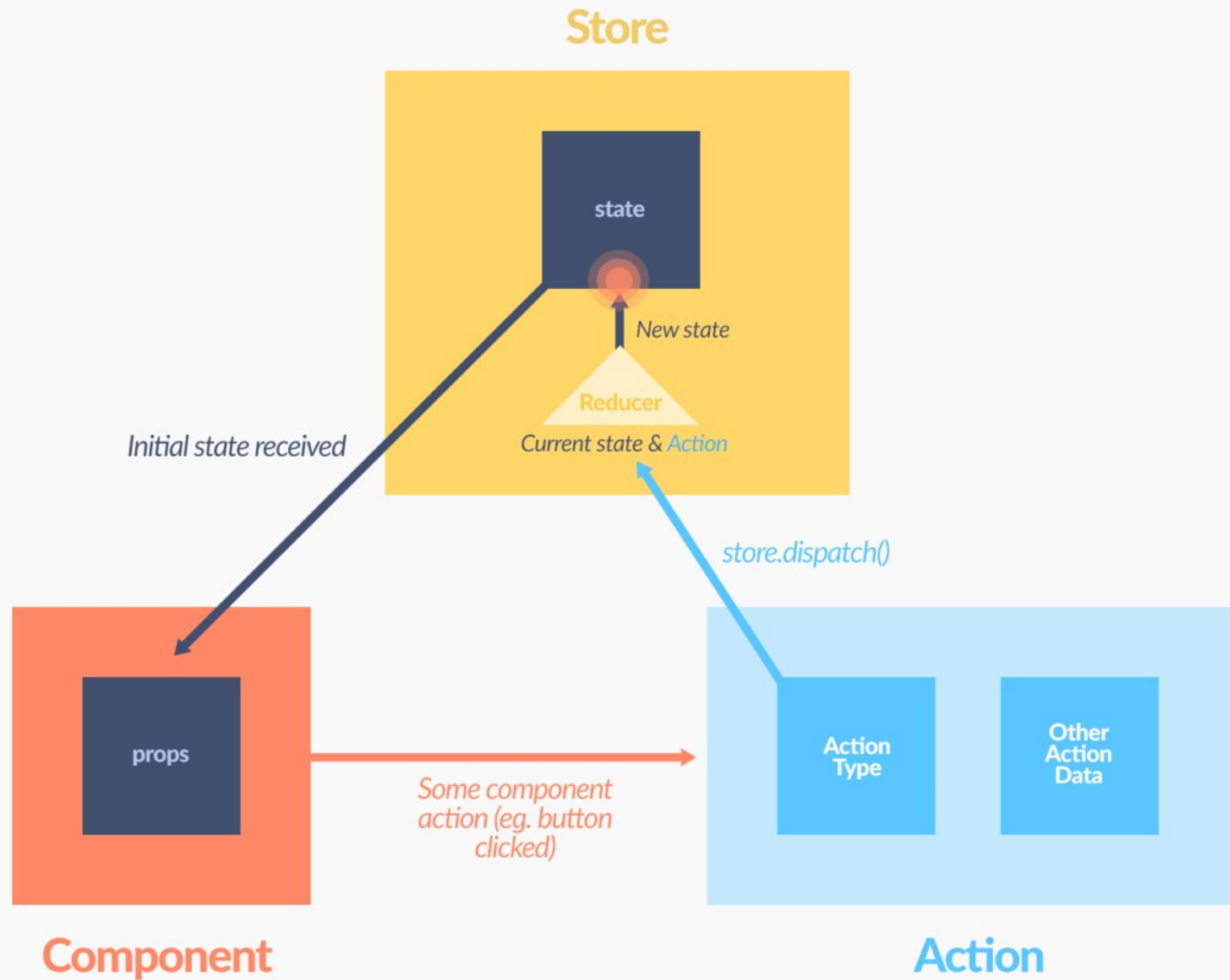
```
function userAddedNewItem(item) {  
  return {  
    type: 'USER_ADDED_TODO',  
    item  
  }  
}
```

```
store.dispatch(userAddedNewItem('buy groceries'));
```



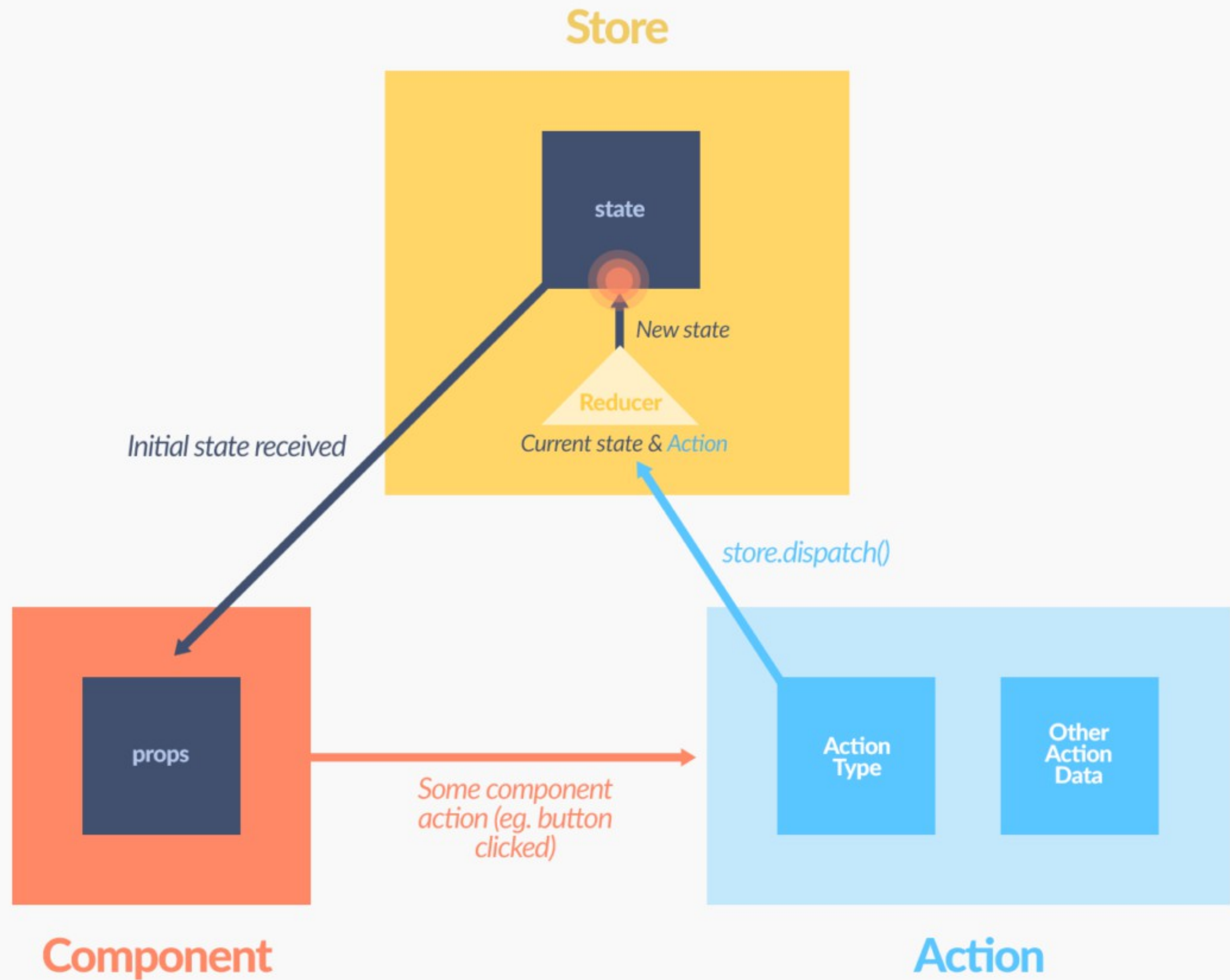


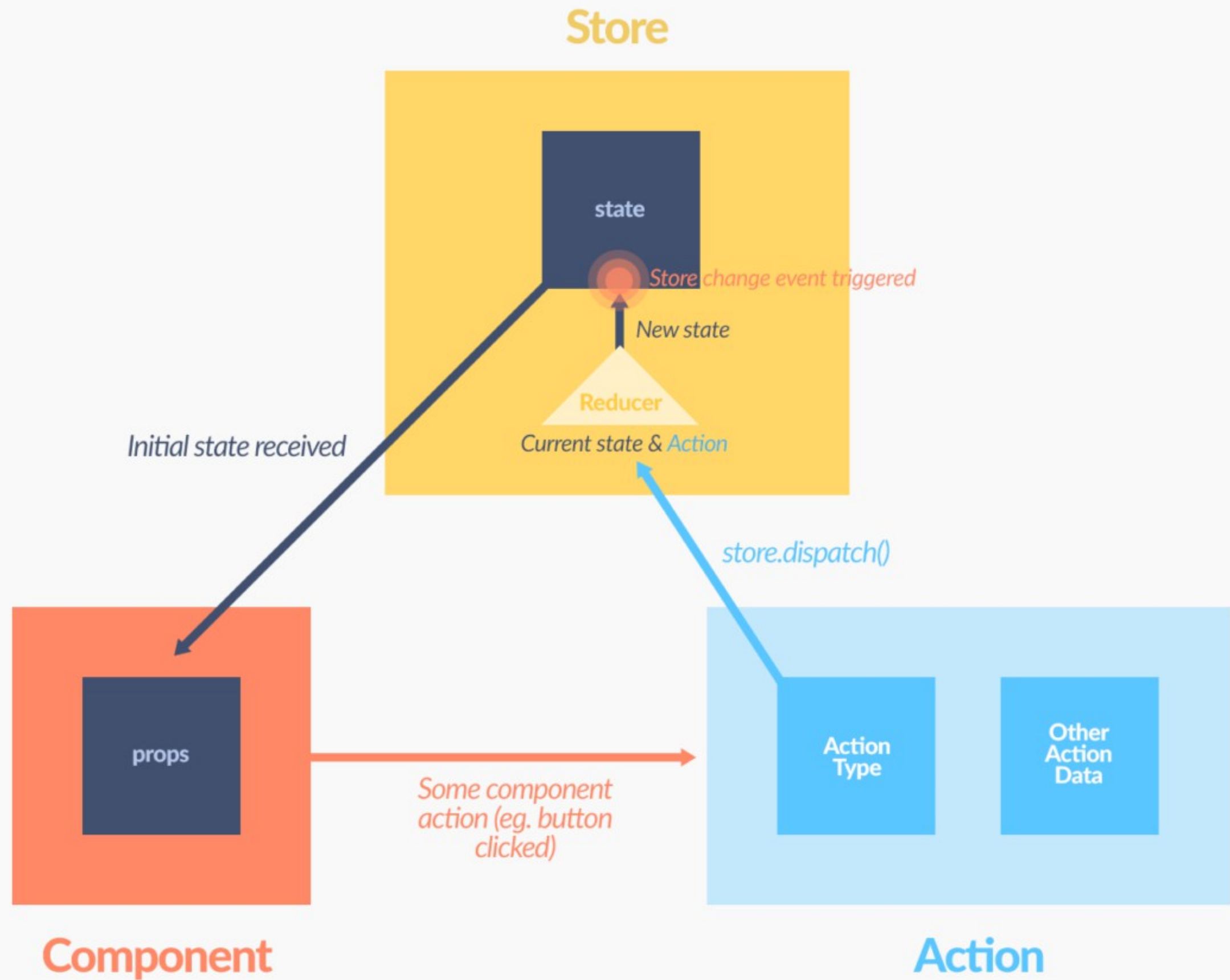


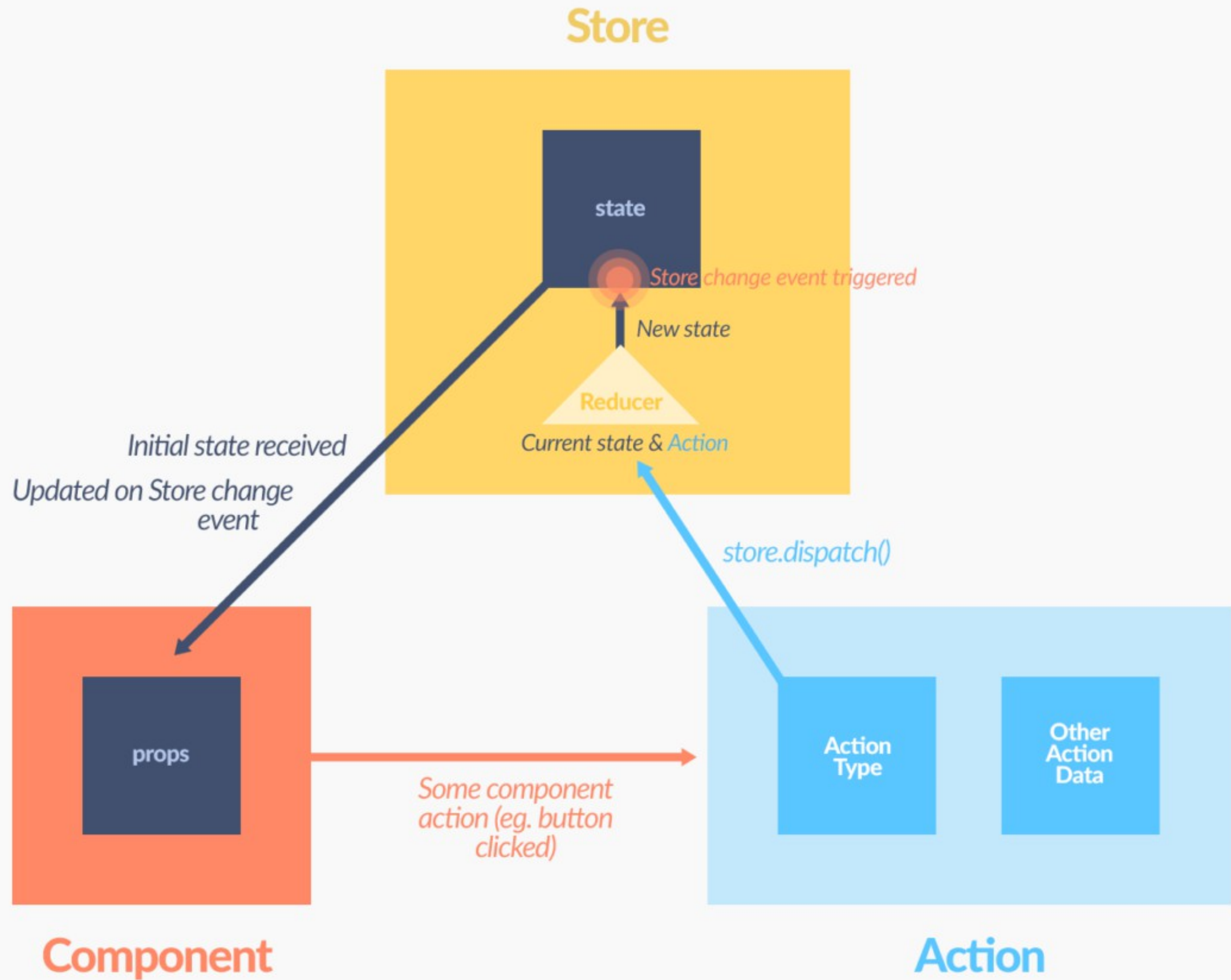


state, action  $\Rightarrow$  state

```
function reducer(state=initialState, action){  
  if (action.type === 'TODO_ITEM_ADDED') {  
    return [...state, action.item];  
  }  
  return state;  
}
```







# todos



*What needs to be done?*



buy groceries

1 item left

All

Active

Completed

Double-click to edit a todo

Created by [petehunt](#)

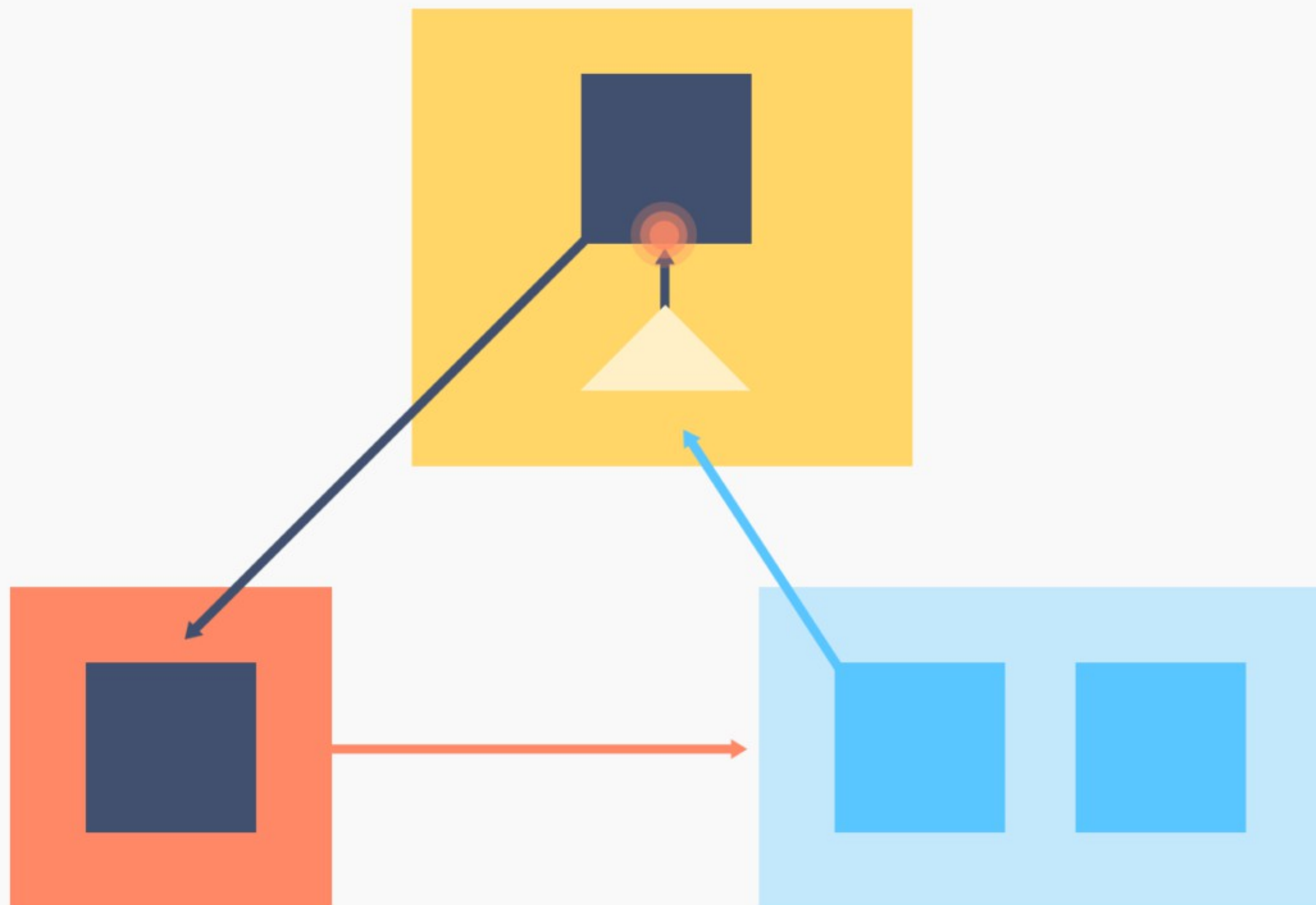
Part of [TodoMVC](#)



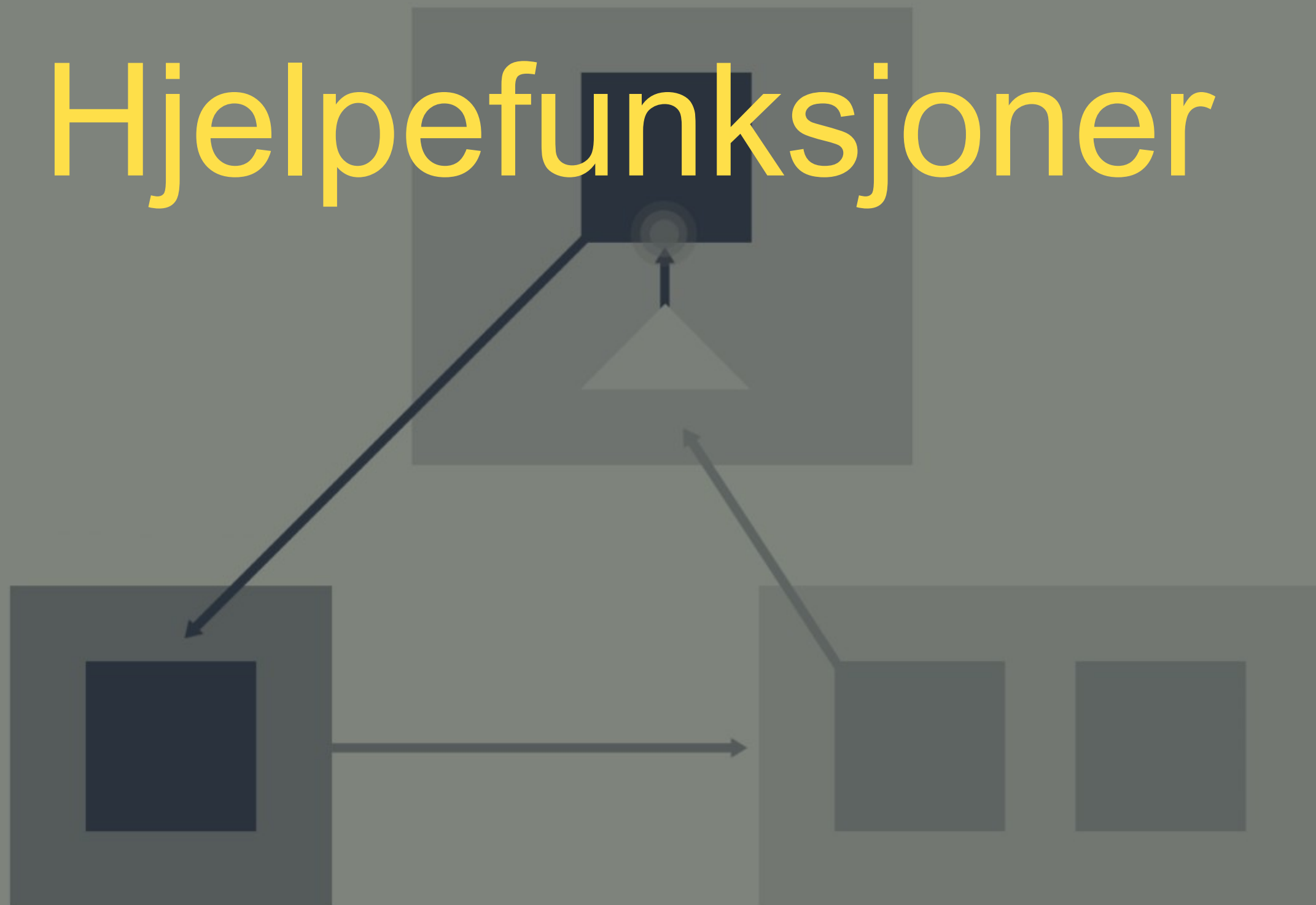
# Koding

<https://github.com/ewendel/redux-workshop>





# Hjelpesfunksjoner



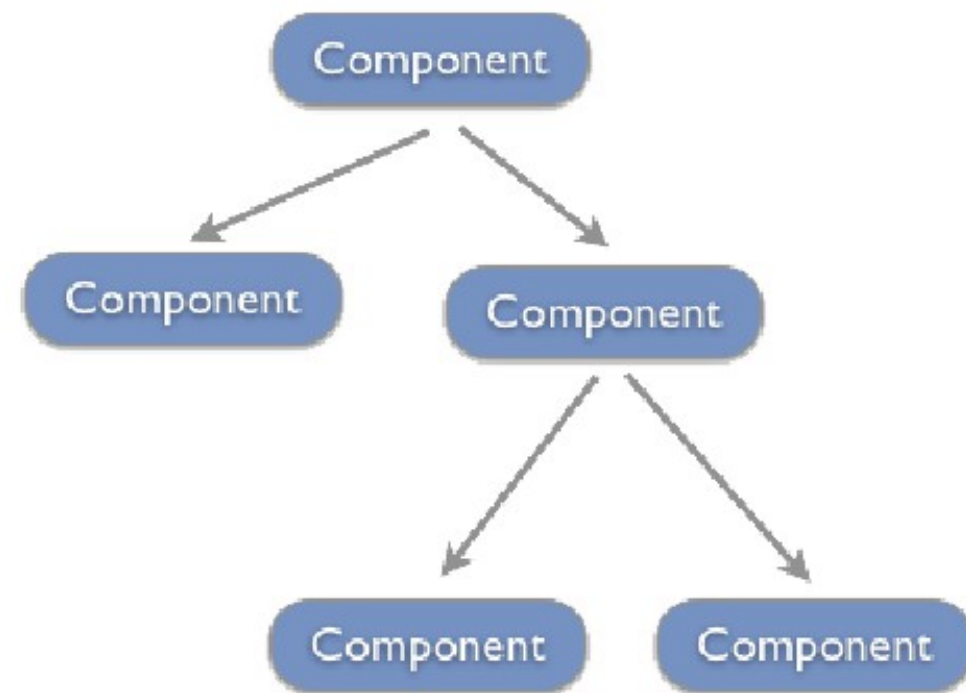
```
import { combineReducers } from 'redux';
```

```
import { connect, Provider } from 'react-redux';
```

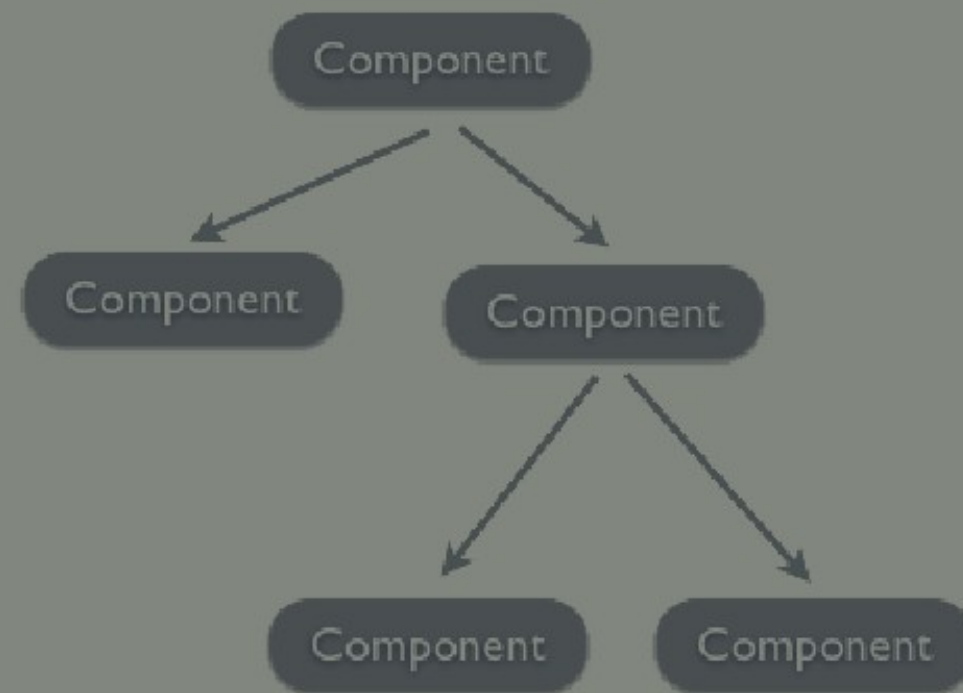
```
combineReducers ({  
  tweets: tweetReducer,  
  viewState: viewStateReducer,  
  route: routeReducer  
})
```

```
const mapStateToProps = state => {  
  return {  
    hasTweets: state.tweets.length > 0,  
    mostRecentTweet: state.tweets[0]  
  }  
}
```

```
connect(mapStateToProps)(MyComponent)
```



# Provider

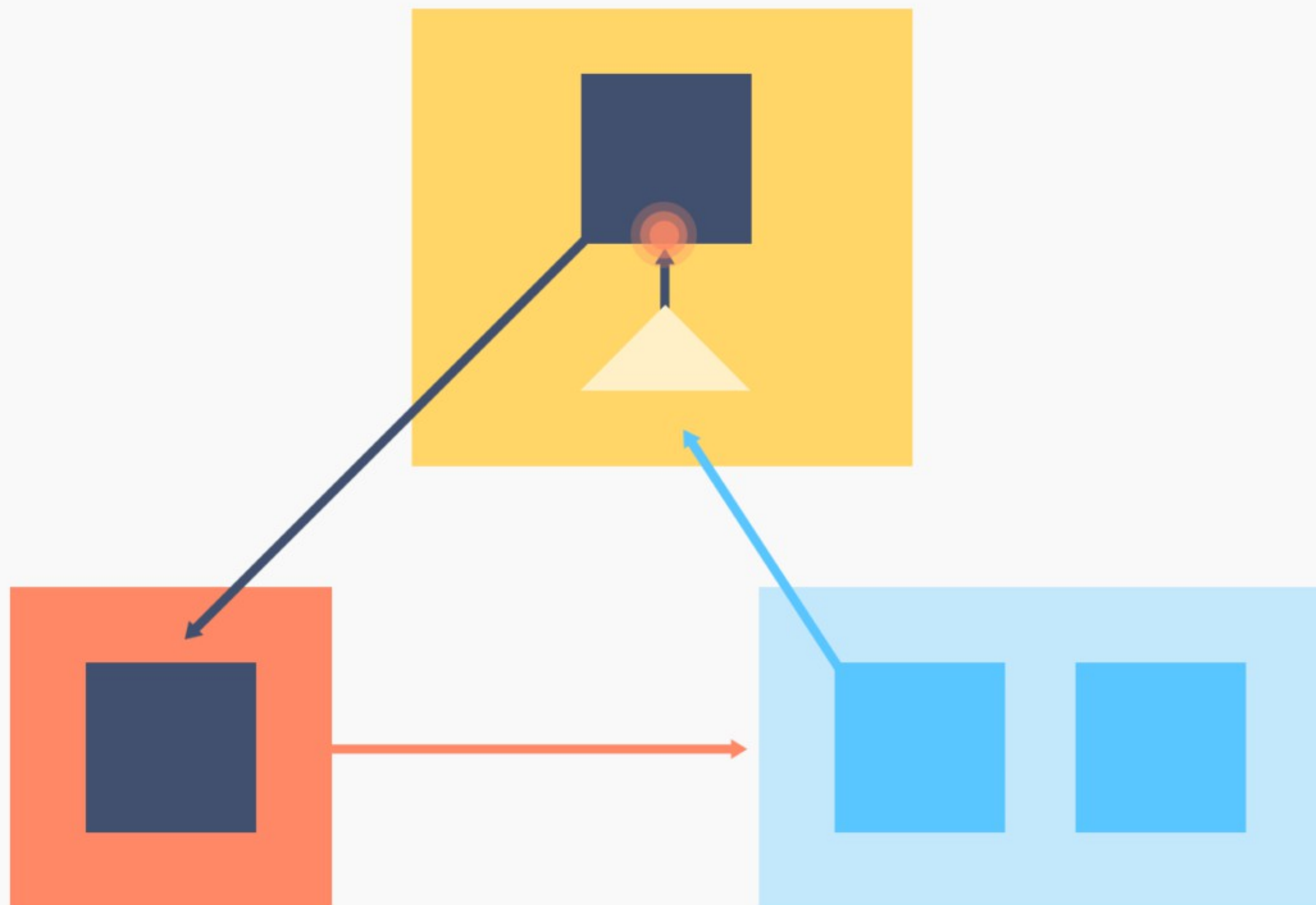




```
<Provider store={store}>
```

```
  <App />
```

```
</Provider>
```



```
function (state = [], action) {  
  switch (action.type) {  
    case TWEET_RECEIVED:  
      return [  
        ...state,  
        action.tweet  
      ];  
    default :  
      return state;  
  }  
}
```