INTRODUCTION & MINI WORKSHOP

# REDUX

## REACT PROBLEMS

## Blue Component

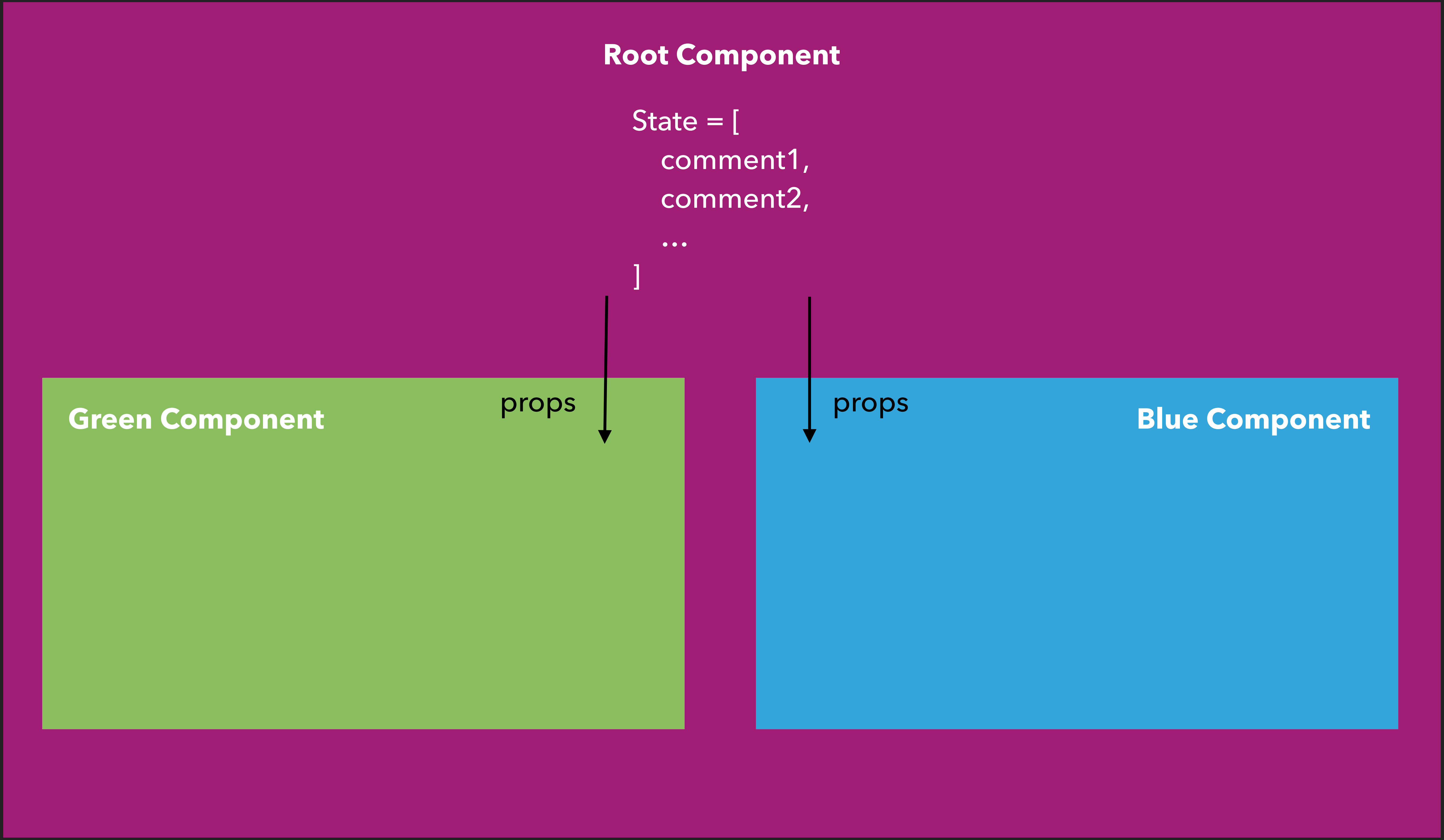
```
State = [
    comment1,
    comment2,
    ...
]
```

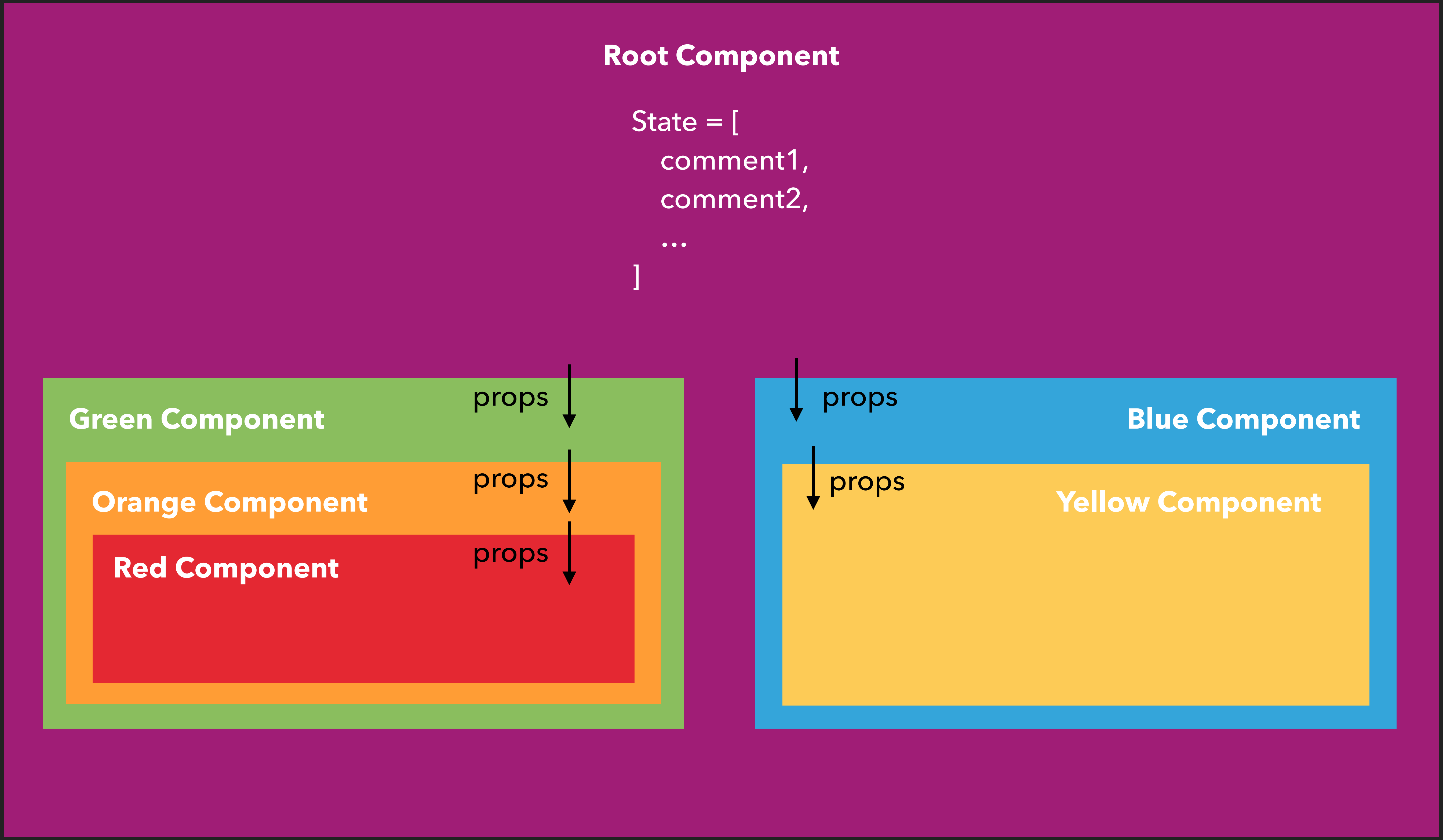
**Green Component**

```
State = [
    comment1,
    comment2,
    ...
]
```

**Blue Component**

```
State = [
    comment1,
    comment2,
    ...
]
```


Root Component
State = [
comment1,
comment2,
...
]
props
props
Green Component
Blue Component

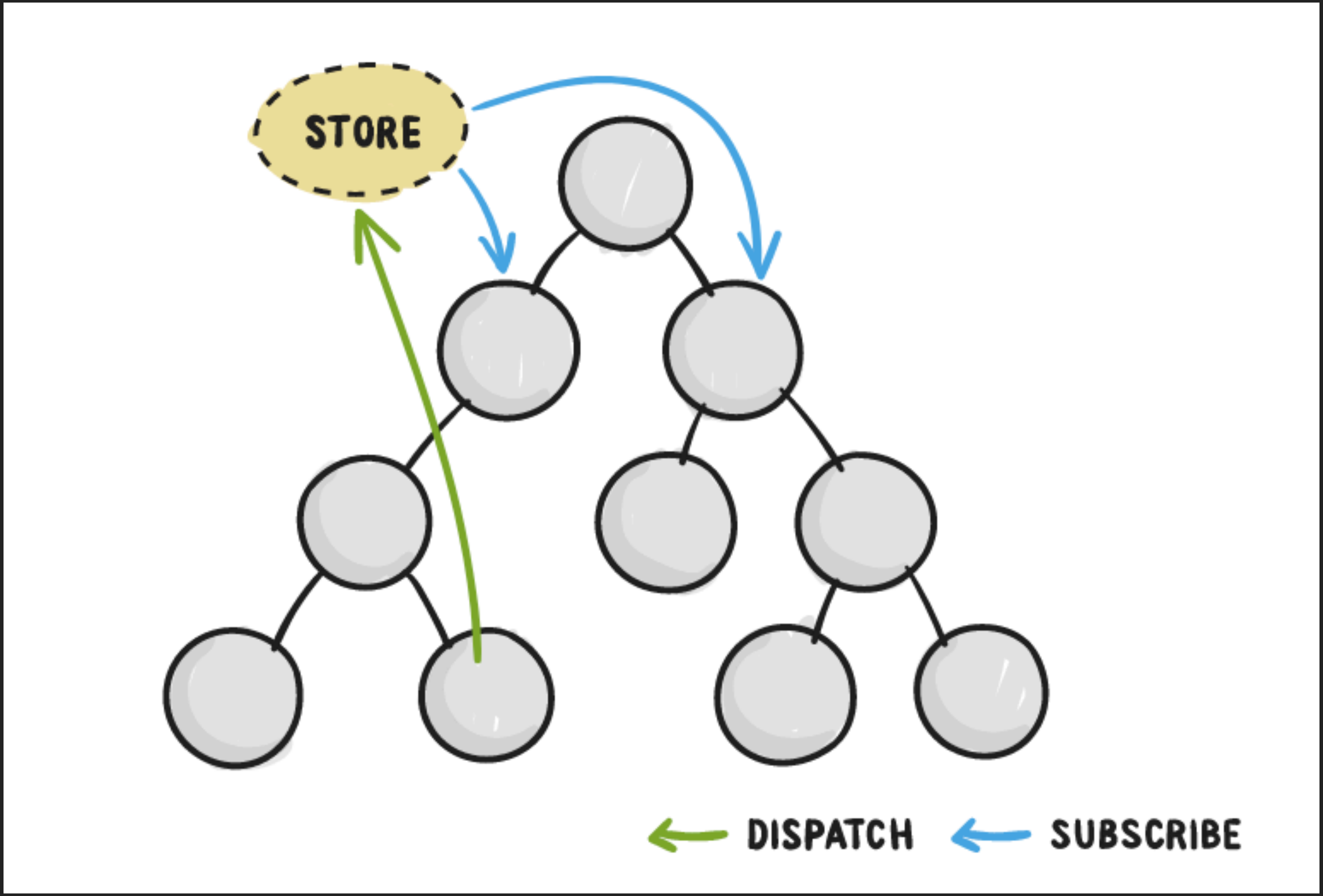

Root Component
State = [
comment1,
comment2,
...
]
Green Component
props
Orange Component
props
Red Component
props
props
Blue Component
props
Yellow Component

## PROBLEMS

- Fat root component
- Single responsibility component
- Reuse component

## Redux

## CORE CONCEPT



https://css-tricks.com/learning-react-redux/

## YOU MIGHT NOT NEED REDUX

- https://medium.com/@dan_abramov/you-might-not-need-redux-be46360cf367

## THREE PRINCIPLES

- Single source of truth
- State is read only
- Change are made with pure functions

- Single source of truth
- State is read only
- Change are made with pure functions

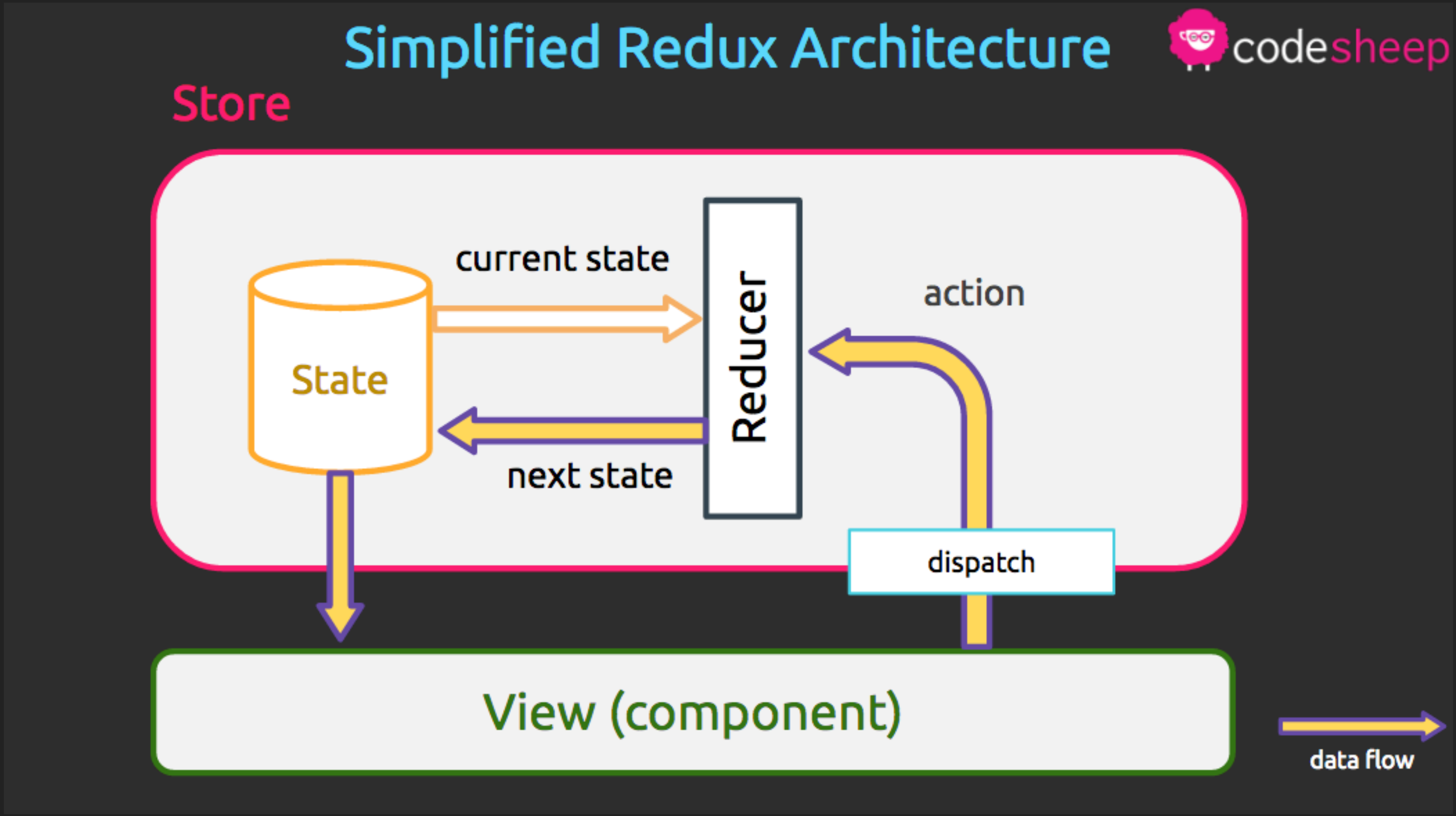
- The state of your whole application is stored in an object tree within a single store.

- Single source of truth
- State is read only
- Change are made with pure functions

- The only way to change the state is to emit an action, an object describing what happened.

- Single source of truth
- State is read only
- Change are made with pure functions

- To specify how the state tree is transformed by actions, you write pure reducers.



codesheep.io

## WORKSHOP

- https://gitlab.thinknet.co.th/kunapot/redux-workshop

- npm run dev
- start at http://localhost:8081

## STEP 1 : CREATE STORE

- Create reducers and initial state
- Create store

## CREATE COUNTER REDUCER

- src/reducers/counterReducer.js

```
import { combineReducers } from 'redux'

const counter = (state = 5, action) => {
  switch (action.type) {
    case "INCREASE":
      return state + 1

    case "DECREASE":
      return state - 1

    default:
      return state
  }
}

const rootReducer = combineReducers({
  counter: counter,
})

export default rootReducer
```

## CREATE STORE

- src/index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import App from './components/App'
import { createStore } from 'redux'
import rootReducer from './reducers/counterReducer'

const store = createStore(rootReducer)

ReactDOM.render(<App />, document.getElementById('root'))
```

## STEP 2 : SUBSCRIBE STATE FROM STORE

- Make store available in component
- Subscribe state in component
- Remove props from App component

## MAKE STORE AVAILABLE IN COMPONENT

- src/index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import { createStore } from 'redux'
import { Provider } from 'react-redux'
import App from './components/App'
import rootReducer from './reducers/counterReducer'

const store = createStore(rootReducer)

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>
, document.getElementById('root'))
```

## SUBSCRIBE STATE IN COMPONENT

- src/components/LeftCounter.js

```
import React, { Component } from 'react'
import { connect } from 'react-redux'

class LeftCounter extends Component {
  render() {
    return (
      <div>
        <button className="button is-primary is-outlined" onClick={this.props.decreaseCounter}>-</button>
        <span className="counter-number">{this.props.counter}</span>
        <button className="button is-primary is-outlined" onClick={this.props.increaseCounter}>+</button>
      </div>
    )
  }
}

const mapStateToProps = state => ({
  counter: state.counter,
})

export default connect(
  mapStateToProps,
)(LeftCounter)
```

## REMOVE PROPS FROM APP COMPONENT

- src/components/App.js

```
<LeftCounter
   increaseCounter={this.increaseCounter}
   decreaseCounter={this.decreaseCounter}
/>
```

## STEP 3 : EMIT ACTION

- Create action creator
- Map action in component
- Remove state and function from App component

## CREATE ACTION CREATOR

- src/actions/counter.js

```
const increaseCounter = () => ({
  type: 'INCREASE',
})

const decreaseCounter = () => ({
  type: 'DECREASE',
})

export {
  increaseCounter,
  decreaseCounter,
}
```

## MAP ACTION IN COMPONENT

- src/components/LeftCounter.js

```
import React, { Component } from 'react'
import { connect } from 'react-redux'
import { increaseCounter, decreaseCounter } from '../actions/counter'

class LeftCounter extends Component {
  render() {
     …
  }
}

const mapStateToProps = state => ({
  counter: state.counter,
})

const mapDispatchToProps = dispatch => ({
  increaseCounter: () => dispatch(increaseCounter()),
  decreaseCounter: () => dispatch(decreaseCounter()),
})

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(LeftCounter)
```

## REMOVE STATE AND ACTION FROM COMPONENT

- src/components/App.js

```
import React, { Component } from 'react'
import LeftCounter from './LeftCounter'
import RightCounter from './RightCounter'

class App extends Component {
  render() {
    return (
      <div className="container section">
        <h1 className="title">Redux Workshop</h1>
        <div className="columns">
          <div className="column">
            <h2 className="subtitle">Left Counter</h2>
            <LeftCounter />
          </div>
          <div className="column">
            <h2 className="subtitle">Right Counter</h2>
            <RightCounter />
          </div>
        </div>
      </div>
    )
  }
}

export default App
```

## STEP 4 : REFACTORING

- Use action type constant
- Split Reducer

## USE ACTION TYPE CONSTANT

- src/constants/actiontypes.js

```
export const INCREASE_COUNTER = 'INCREASE_COUNTER'
export const DECREASE_COUNTER = 'DECREASE_COUNTER'
```

## USE ACTION TYPE CONSTANT

- src/reducers/counterReducer.js

```
import { INCREASE_COUNTER, DECREASE_COUNTER } from '../constants/actiontypes'

const counter = (state = 5, action) => {
  switch (action.type) {
    case INCREASE_COUNTER:
      return state + 1

    case DECREASE_COUNTER:
      return state - 1

    default:
      return state
  }
}

export default counter
```

## USE ACTION TYPE CONSTANT

- src/actions/counter.js

```
import { INCREASE_COUNTER, DECREASE_COUNTER } from '../constants/actiontypes'

const increaseCounter = () => ({
  type: INCREASE_COUNTER,
})

const decreaseCounter = () => ({
  type: DECREASE_COUNTER,
})

export {
  increaseCounter,
  decreaseCounter,
}
```

## SPLIT REDUCER

- src/reducers/index.js

```
import { combineReducers } from 'redux'
import counterReducer from './counterReducer'

const rootReducer = combineReducers({
  counter: counterReducer,
})

export default rootReducer
```

## SPLIT REDUCER

- src/reducers/counterReducer.js

```
import { INCREASE_COUNTER, DECREASE_COUNTER } from '../constants/actiontypes'

const counter = (state = 5, action) => {
  switch (action.type) {
    case INCREASE_COUNTER:
      return state + 1

    case DECREASE_COUNTER:
      return state - 1

    default:
      return state
  }
}

export default counter
```

## SPLIT REDUCER

- src/index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import { createStore } from 'redux'
import { Provider } from 'react-redux'
import App from './components/App'
import rootReducer from './reducers'

const store = createStore(rootReducer)

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>
, document.getElementById('root'))
```

## USEFUL ASSETS

- Redux devtools (redux debugging tool)
- Unit Testing (next session)