INTRODUCTION TO REDUX

Nacho Martín
@nacmartin

React Native Munich Meetup
January 2017

Nacho Martin

I write code at Limenius.

We build tailor-made projects,
and provide consultancy
and formation.

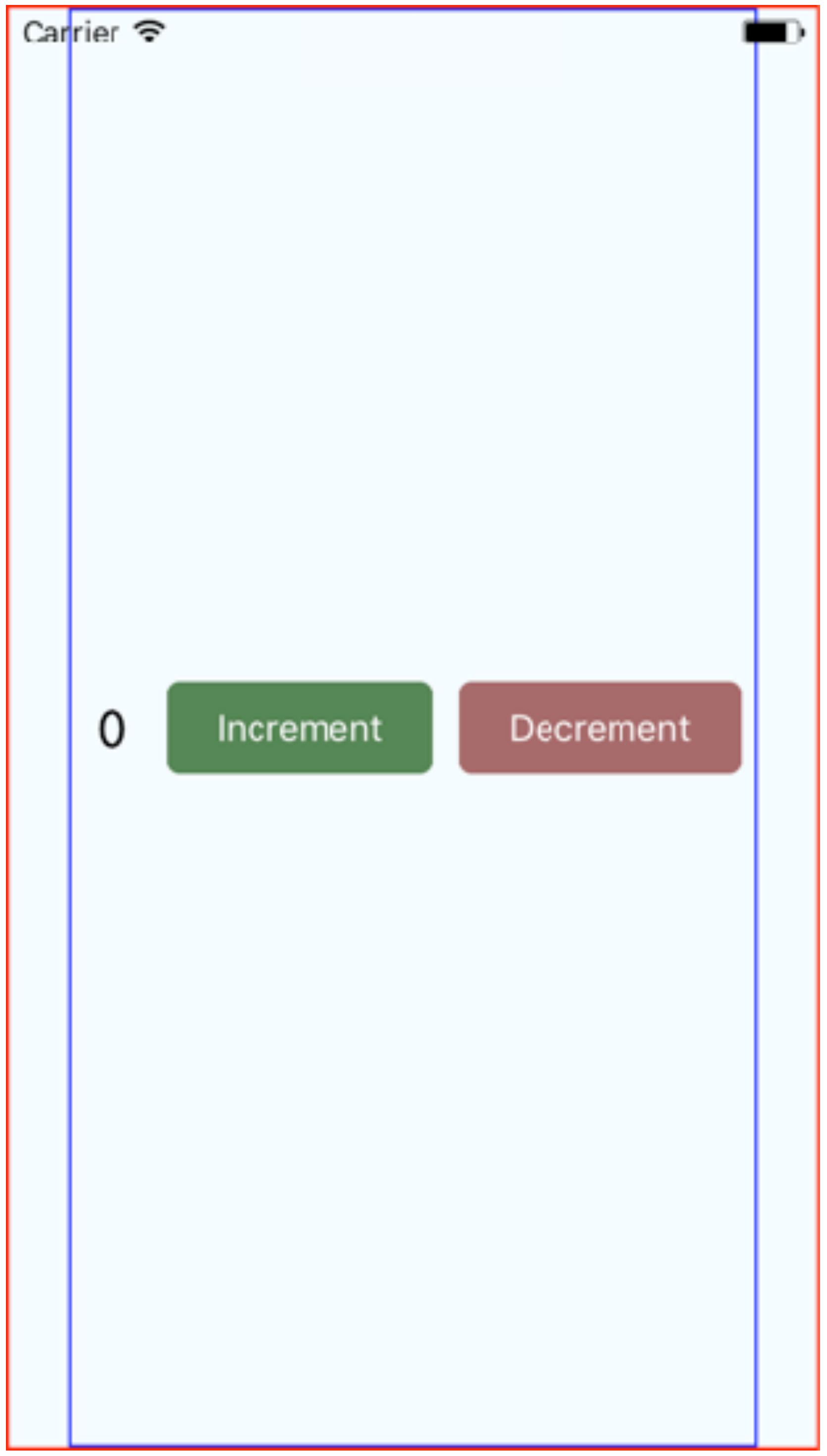
We are very happy with React and React Native.

What is Redux?

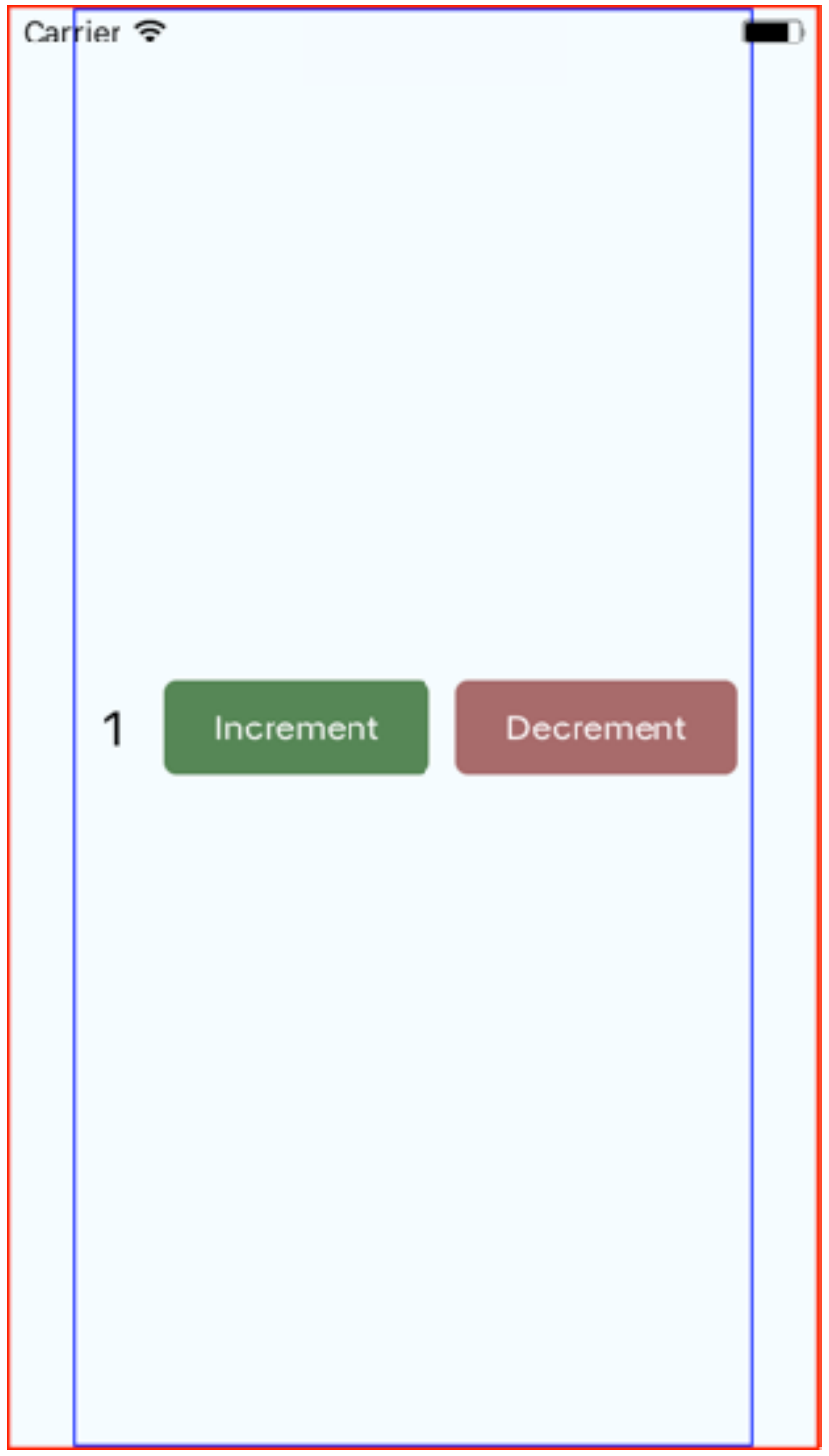
- State container
- Created by Dan Abramov
- Inspired by Flux and Elm
- Can be used without React

What problem does Redux solve?

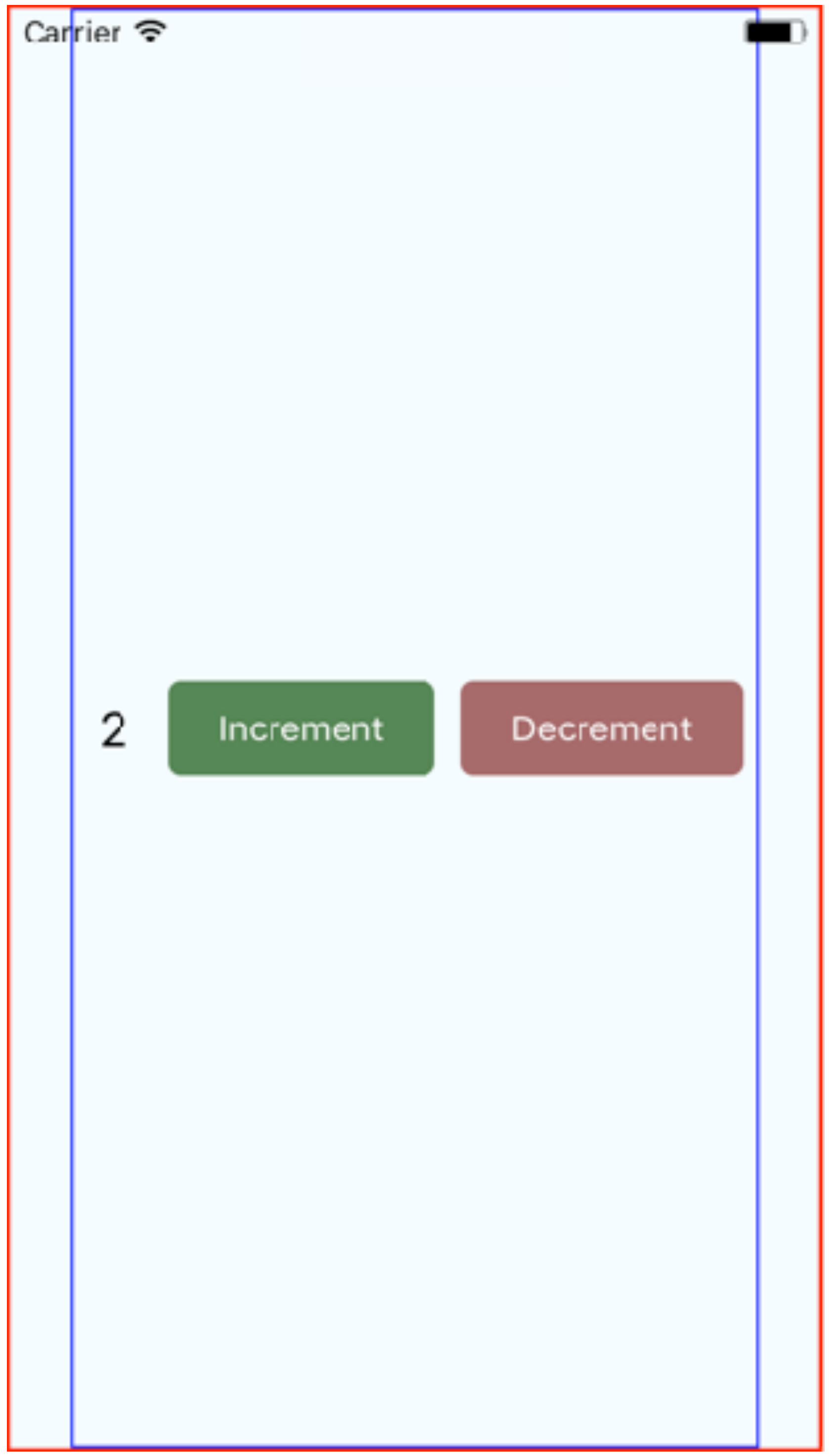
A simple component



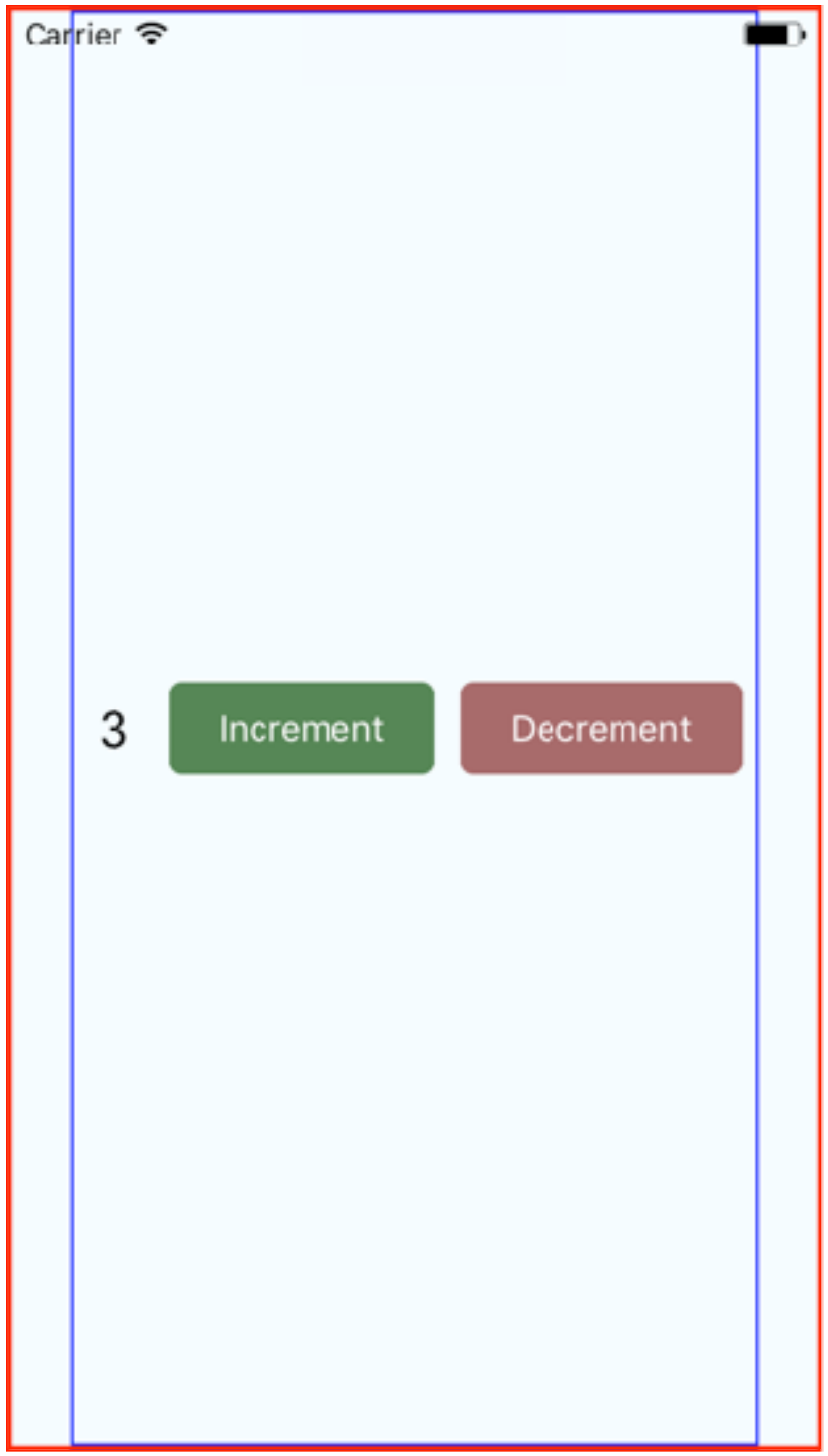
A simple component



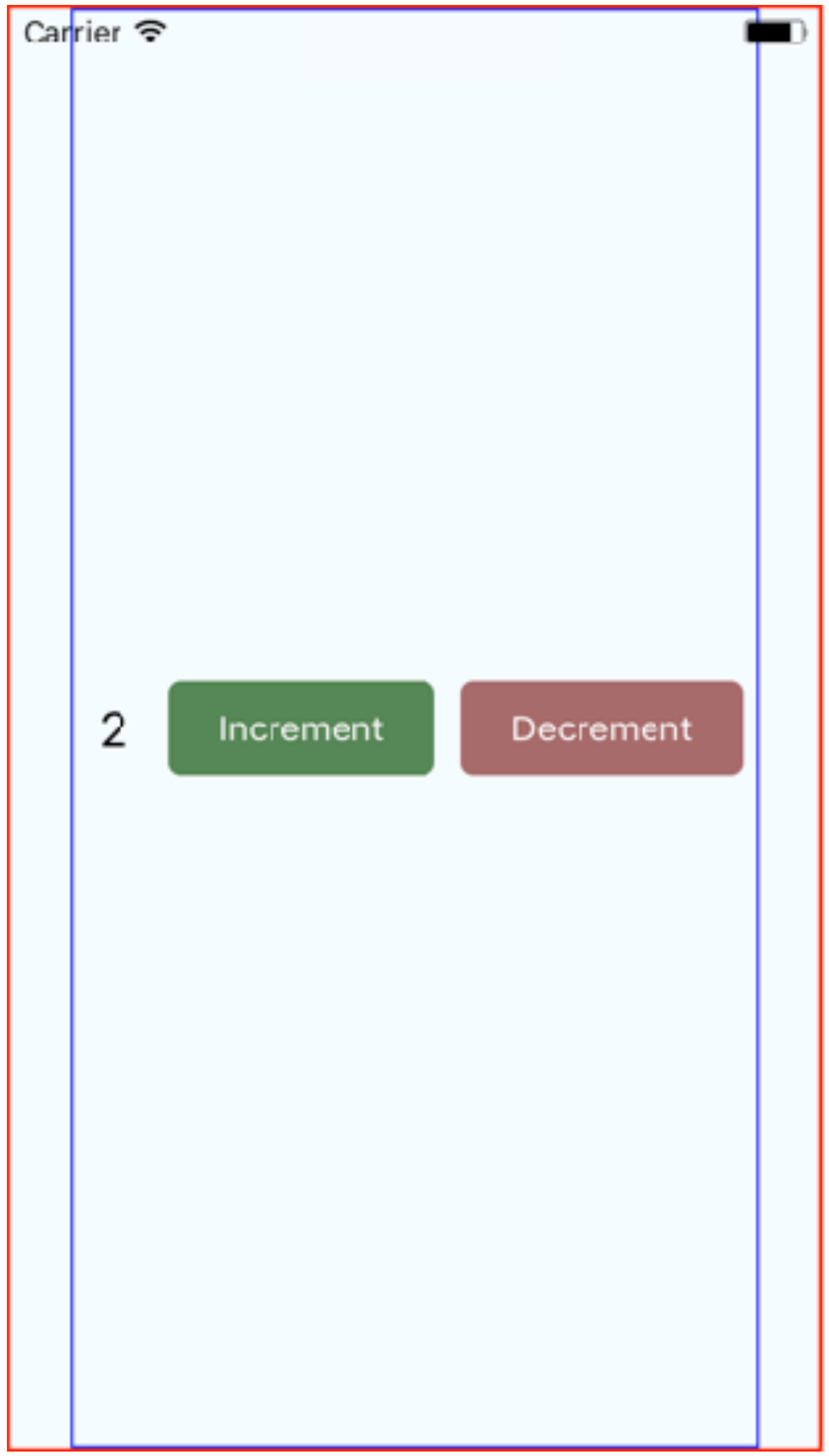
A simple component



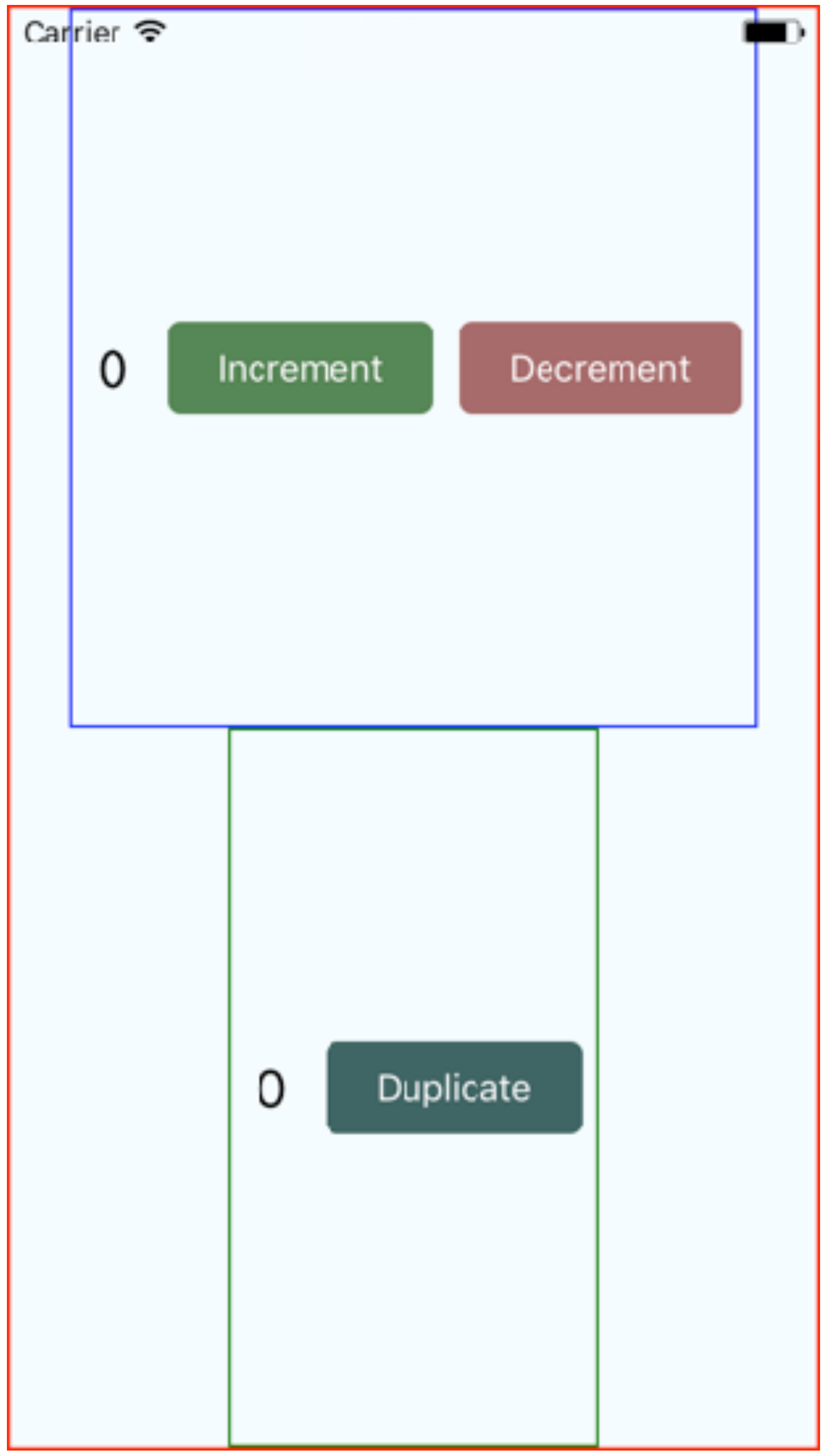
A simple component



A simple component



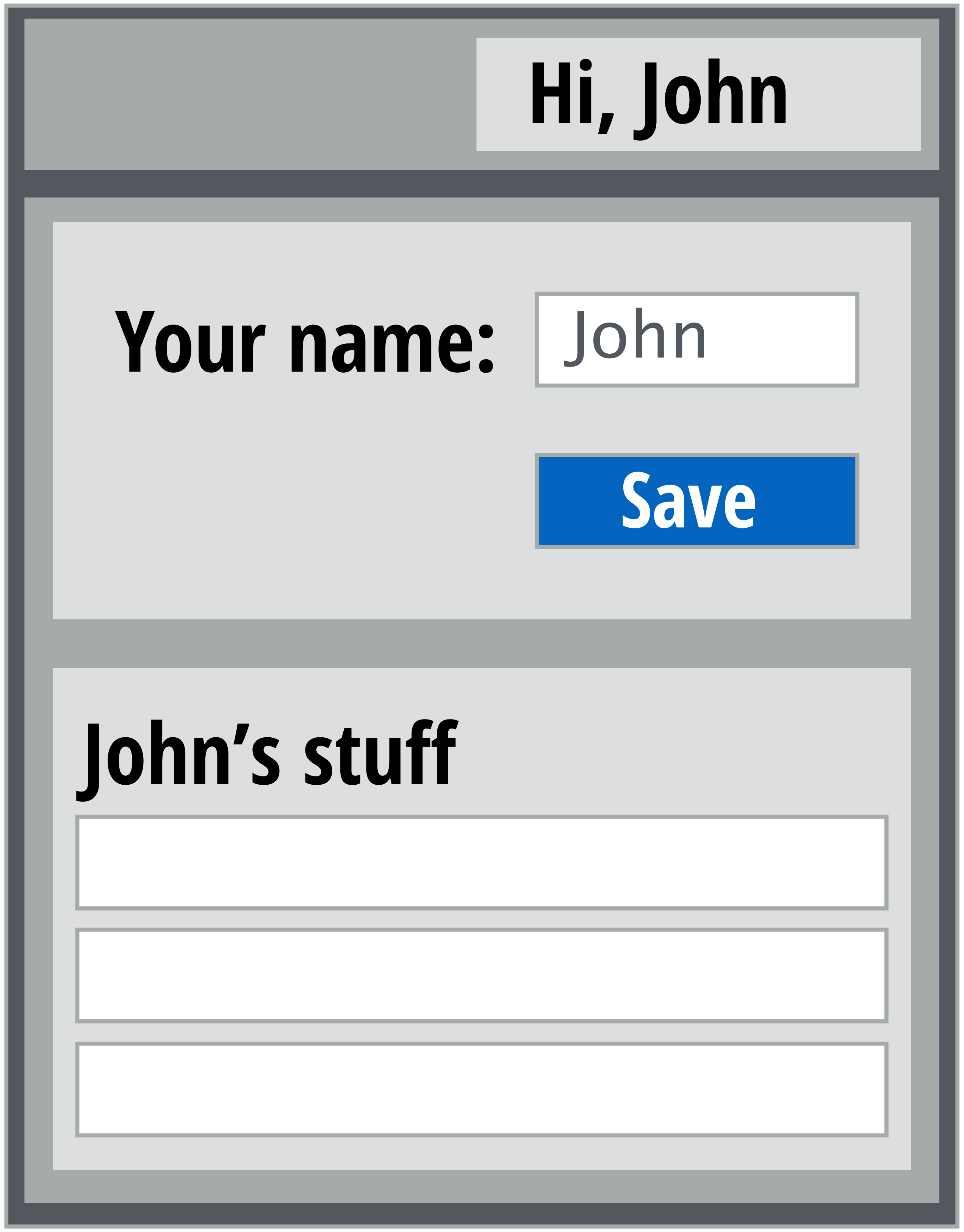
We have a new requirement



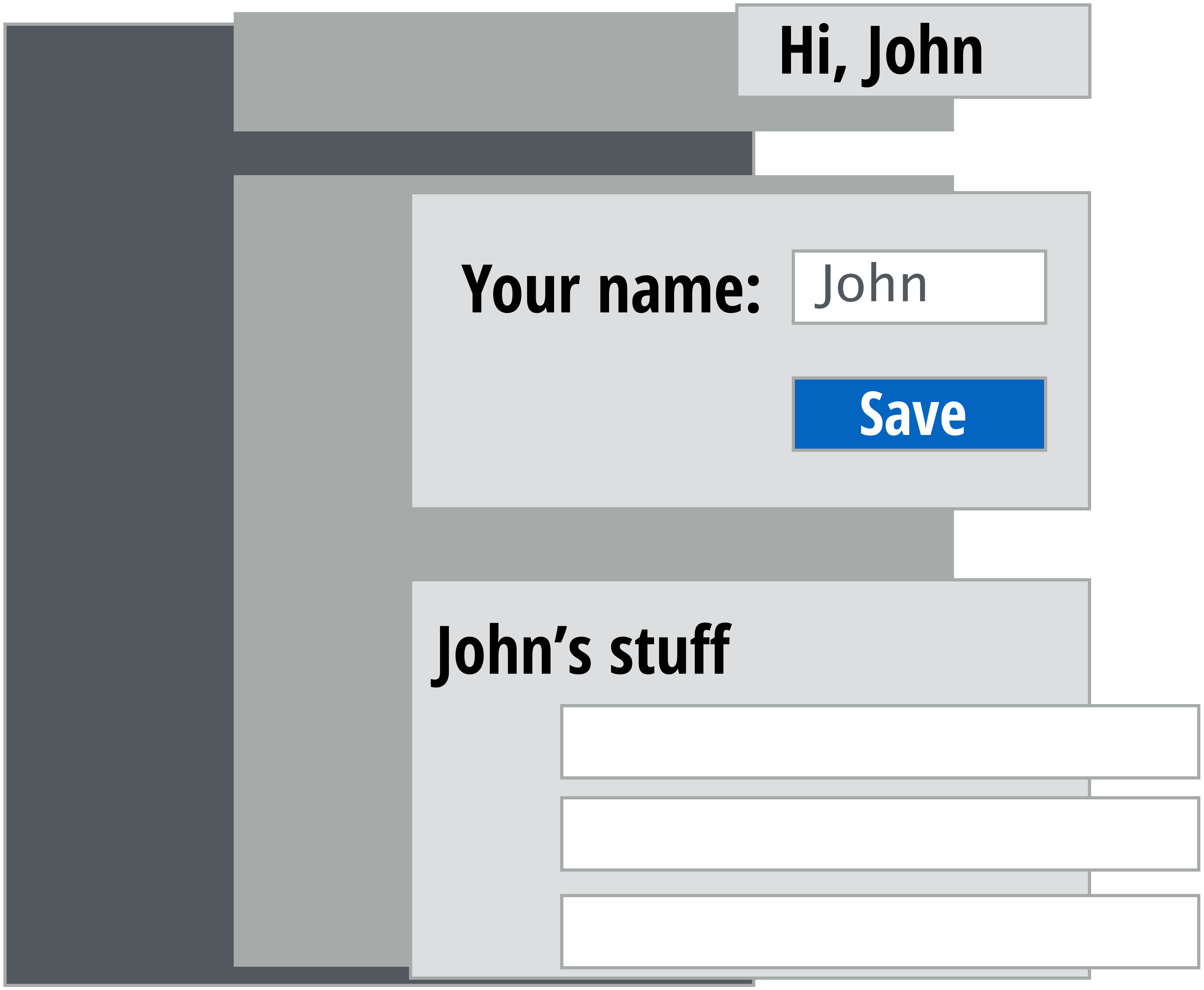
Naive approach 1: pass props

Intermediate components receive props and pass them to their children **without using them**.

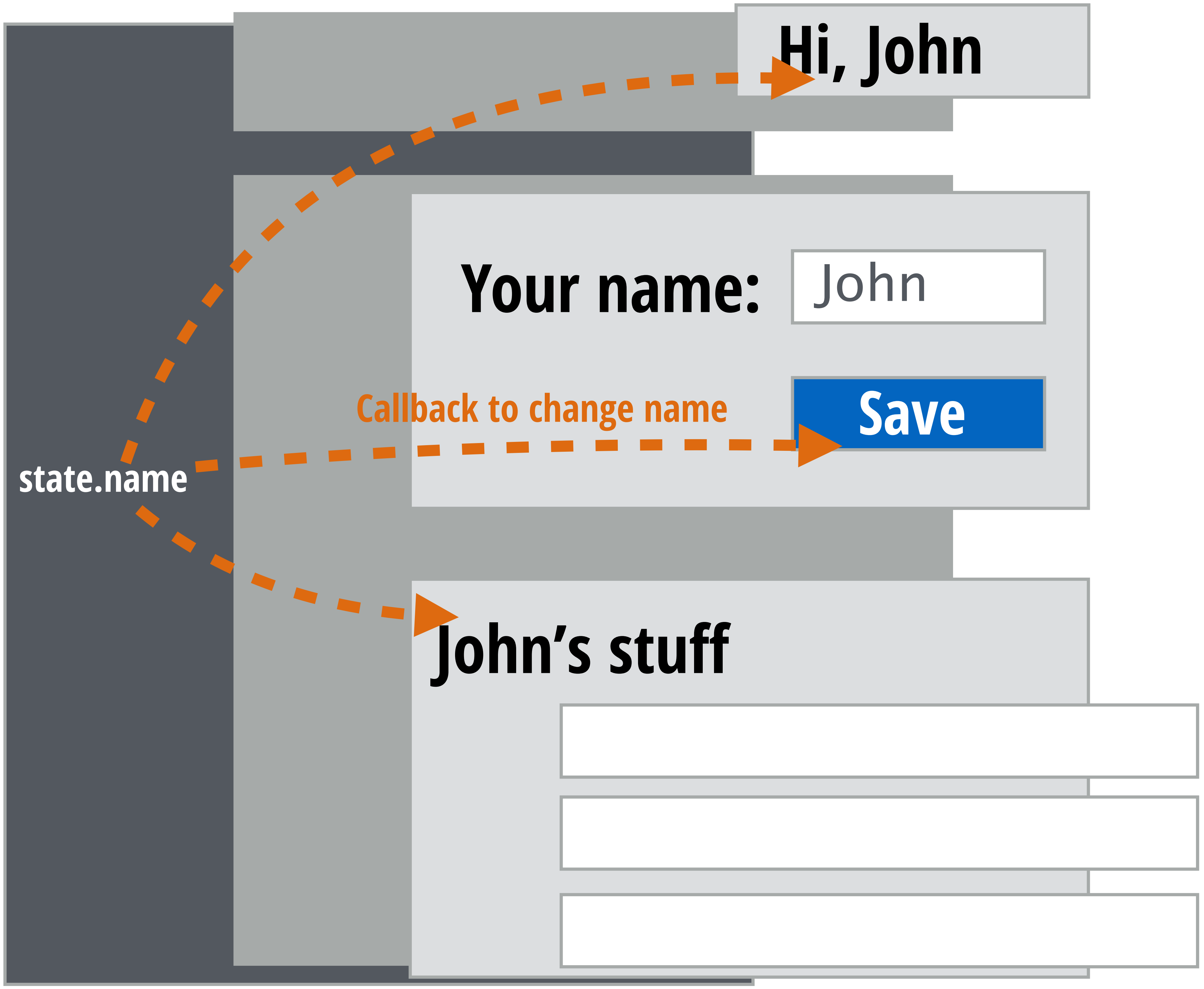
Cauliflower



Cauliflower



Cauliflower



Naive approach 1: pass props

Intermediate components receive props and pass them to their children **without using them**.

Naive approach 1: pass props

Intermediate components receive props and pass them to their children **without using them**.

Problem: Messy

Difficult to understand the flow of data.

Naive approach 2: use context

Context is a way to pass data down to the tree, without doing it manually.

Naive approach 2: use context

Context is a way to pass data down to the tree, without doing it manually.

Problem: Dangerous

Context API is experimental

But Redux uses the Context API internally. Isn’t it a problem?

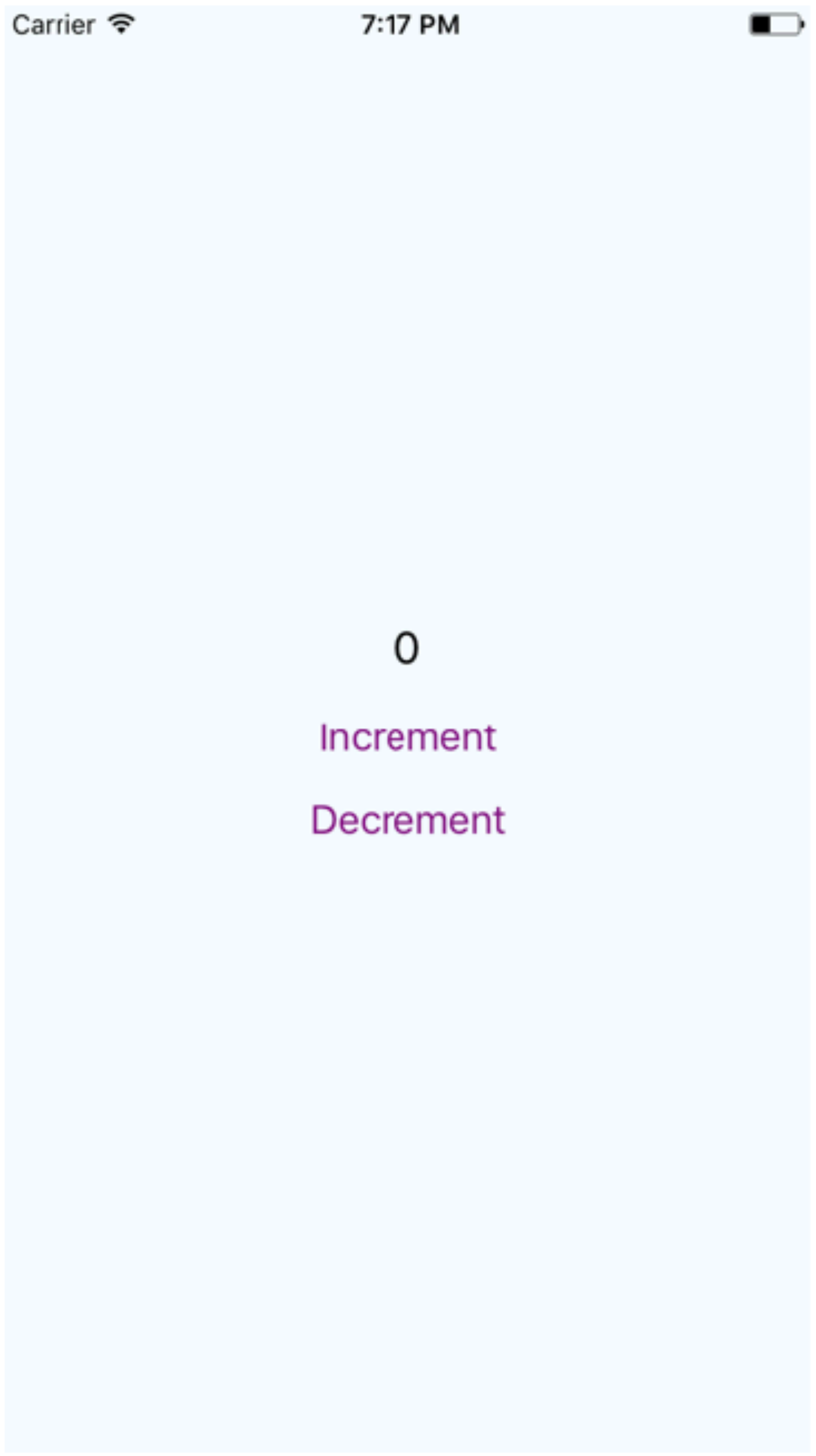
But Redux uses the Context API internally. Isn’t it a problem?

It is better to use libs that use it than use it directly.

The Redux way

Example

```
export default class Counter extends Component {
    constructor(props) {
        super(props);
        this.state = {counter: 0};
    }

    increment() {
        this.setState({counter: this.state.counter + 1});
    }

    decrement() {
        this.setState({counter: this.state.counter - 1});
    }

    render() {
        return (
            <View style={styles.container}>
                <Text style={styles.welcome}>
                    {this.state.counter}
                </Text>
                <Button
                    onPress={this.increment.bind(this)}
                    title="Increment"
                />
                <Button
                    onPress={this.decrement.bind(this)}
                    title="Decrement"
                />
            </View>
        );
    }
}
```



```
increment() {
    this.setState({counter: this.state.counter + 1});
}

decrement() {
    this.setState({counter: this.state.counter - 1});
}
```

What about this?

```
dispatch(action) {
    this.setState(reducer(this.state, action));
}

increment() {
    this.dispatch({type: 'INCREMENT'});
}

decrement() {
    this.dispatch({type: 'DECREMENT'});
}
```

What about this?

```
dispatch(action) {
    this.setState(reducer(this.state, action));
}

increment() {
    this.dispatch({type: 'INCREMENT'});
}

decrement() {
    this.dispatch({type: 'DECREMENT'});
}
```

Action

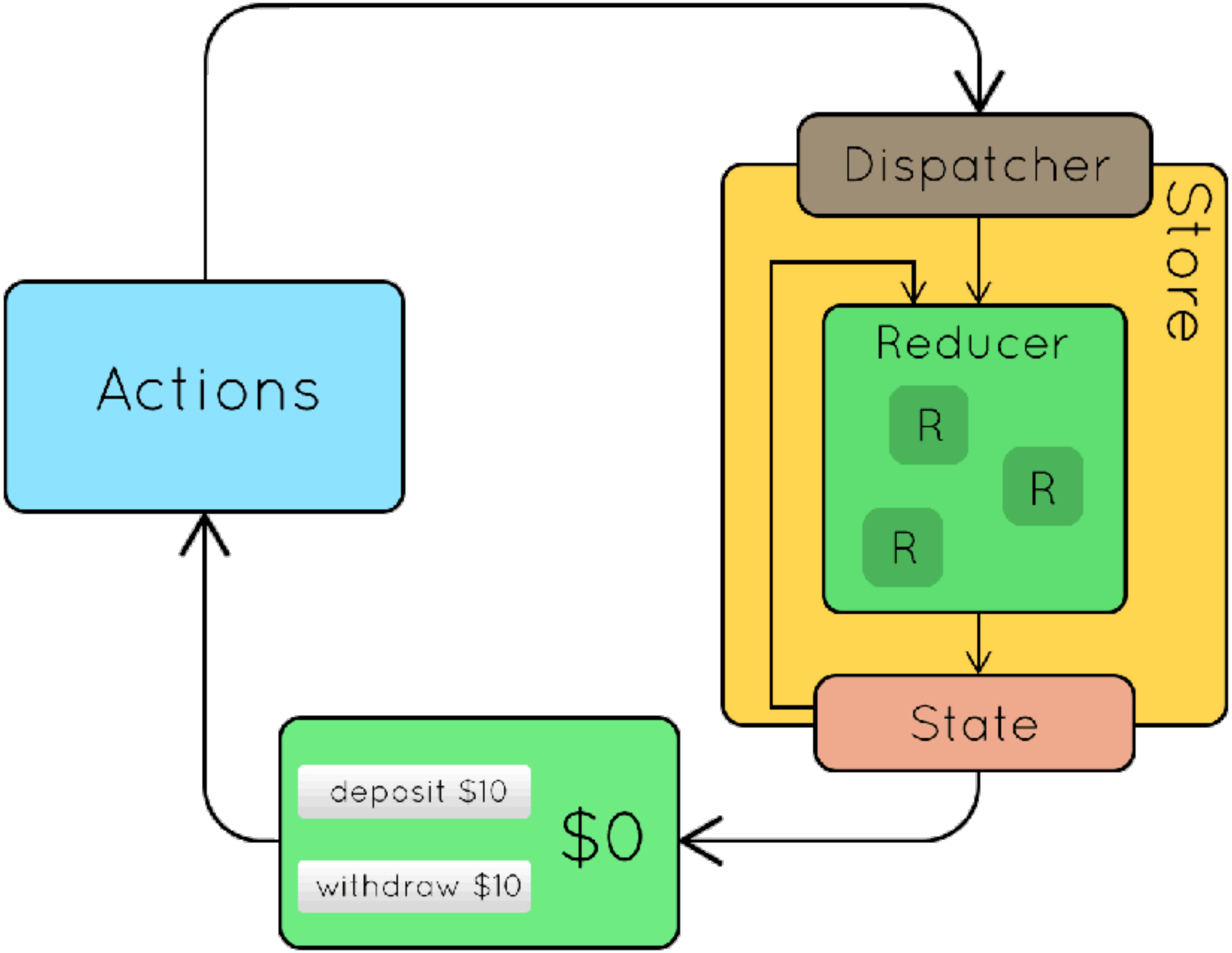
What about this?

```
dispatch(action) {
    this.setState(reducer(this.state, action));
}

increment() {
    this.dispatch({type: 'INCREMENT'});
}

decrement() {
    this.dispatch({type: 'DECREMENT'});
}
```

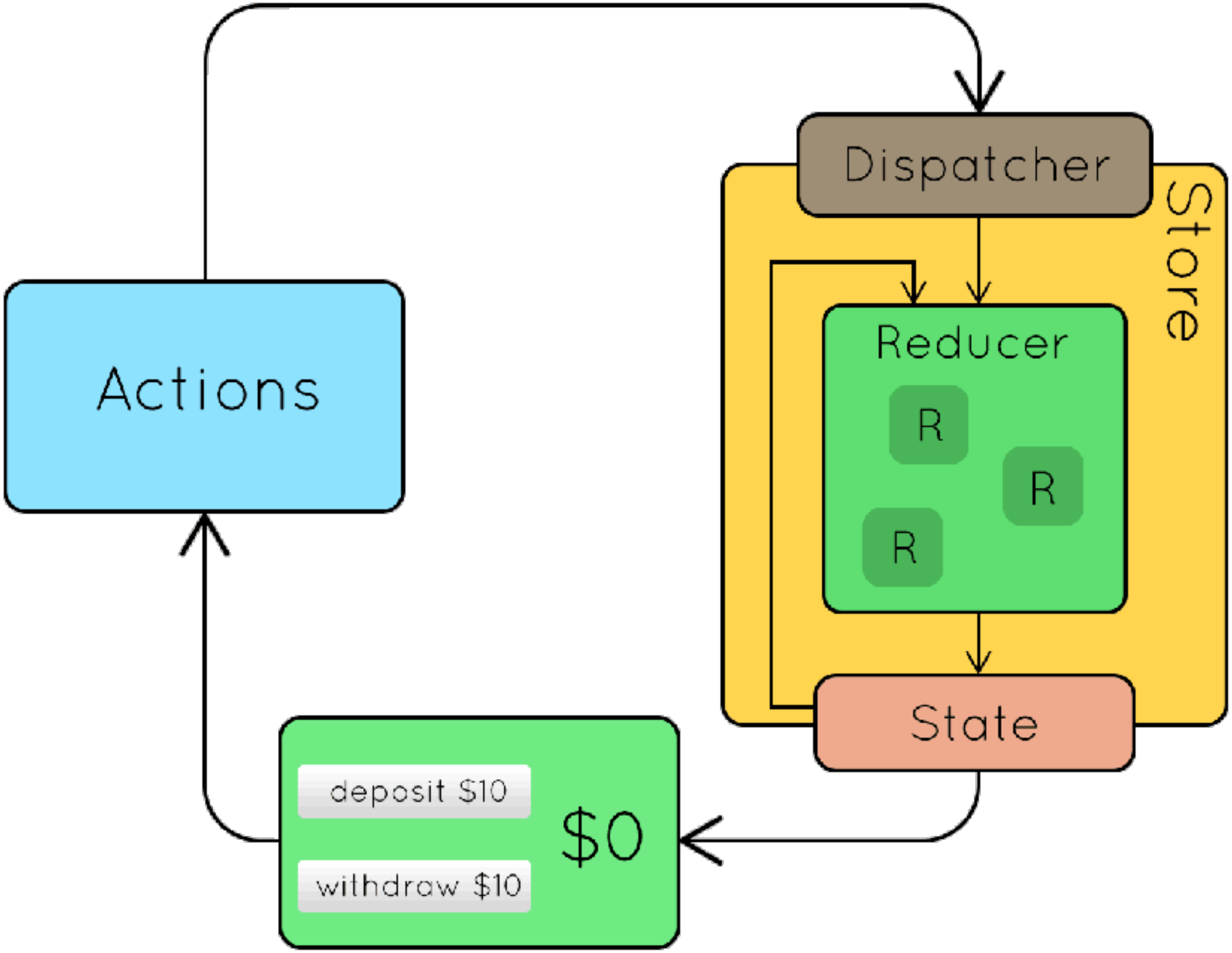
Action

```
const reducer = (state = { counter: 0 }, action) => {
    switch (action.type) {
        case 'INCREMENT':
            return { counter: state.counter + 1 };
        case 'DECREMENT':
            return { counter: state.counter - 1 };
        default:
            return state;
    }
}
```

Redux in a picture



Redux in a picture



Install redux

```
npm install --save redux react-redux
```

Actions

Only **source of information** from views to change state

Plain **JS objects**

Must have a **type**

```
const increment = {
    type: 'INCREMENT'
}
```

Actions

Only **source of information** from views to change state

Plain **JS objects**

Must have a **type**

```
const todoAdd = {
    type: 'TODO_ADD',
    text: 'Buy some milk',
    deadline: '15-01-2017 19:00h'
}
```

Objects can be as complex as needed, but **keep it simple**

Reducers

```
export default reducer = (state = { counter: 0 }, action) => {
    switch (action.type) {
        case 'INCREMENT':
            return {
                ...state,
                counter: state.counter + 1
            };
        case 'DECREMENT':
            return {
                ...state,
                counter: state.counter - 1
            };
        case 'DUPLICATE':
            return {
                ...state,
                counter: state.counter * 2
            };
        default:
            return state;
    }
}
```

From previous state and action produce next state

Store

```
import { Provider } from 'react-redux'
import { createStore } from 'redux'
import counterReducer from './redux/reducer';

let store = createStore(counterReducer)

export default class CounterApp  extends Component {
    render() {
        return (
            <Provider store={store}>
               <Counter/>
               <Multiplier/>
            </Provider>
        );
    }
}
```

Connect & dispatch

```
import { connect } from 'react-redux'

const Counter = (props) => {
    return (
        <View style={styles.counter}>
            <Text>
                {props.counter}
            </Text>
            <Button
                onPress={() => {props.dispatch({type: 'INCREMENT'})}}
                title="Increment"
            />
            <Button
                onPress={() => {props.dispatch({type: 'DECREMENT'})}}
                title="Decrement"
            />
        </View>
    );
}

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

export default connect(mapStateToProps)(Counter);
```

Connect & dispatch

```
import { connect } from 'react-redux'

const Counter = (props) => {
    return (
        <View style={styles.counter}>
            <Text>
                {props.counter}
            </Text>
            <Button
                onPress={() => {props.dispatch({type: 'INCREMENT'})}}
                title="Increment"
            />
            <Button
                onPress={() => {props.dispatch({type: 'DECREMENT'})}}
                title="Decrement"
            />
        </View>
    );
}
const mapStateToProps = (state) => {
    return {
        counter: state.counter,
    }
};

export default connect(mapStateToProps)(Counter);
```

Export the connected component

Connect & dispatch

```
import { connect } from 'react-redux'

const Counter = (props) => {
    return (
        <View style={styles.counter}>
            <Text>
                {props.counter}
            </Text>
            <Button
                onPress={() => {props.dispatch({type: 'INCREMENT'})}}
                title="Increment"
            />
            <Button
                onPress={() => {props.dispatch({type: 'DECREMENT'})}}
                title="Decrement"
            />
        </View>
    );
}
```

Counter available via props

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

export default connect(mapStateToProps)(Counter);
```

Export the connected component

Connect & dispatch

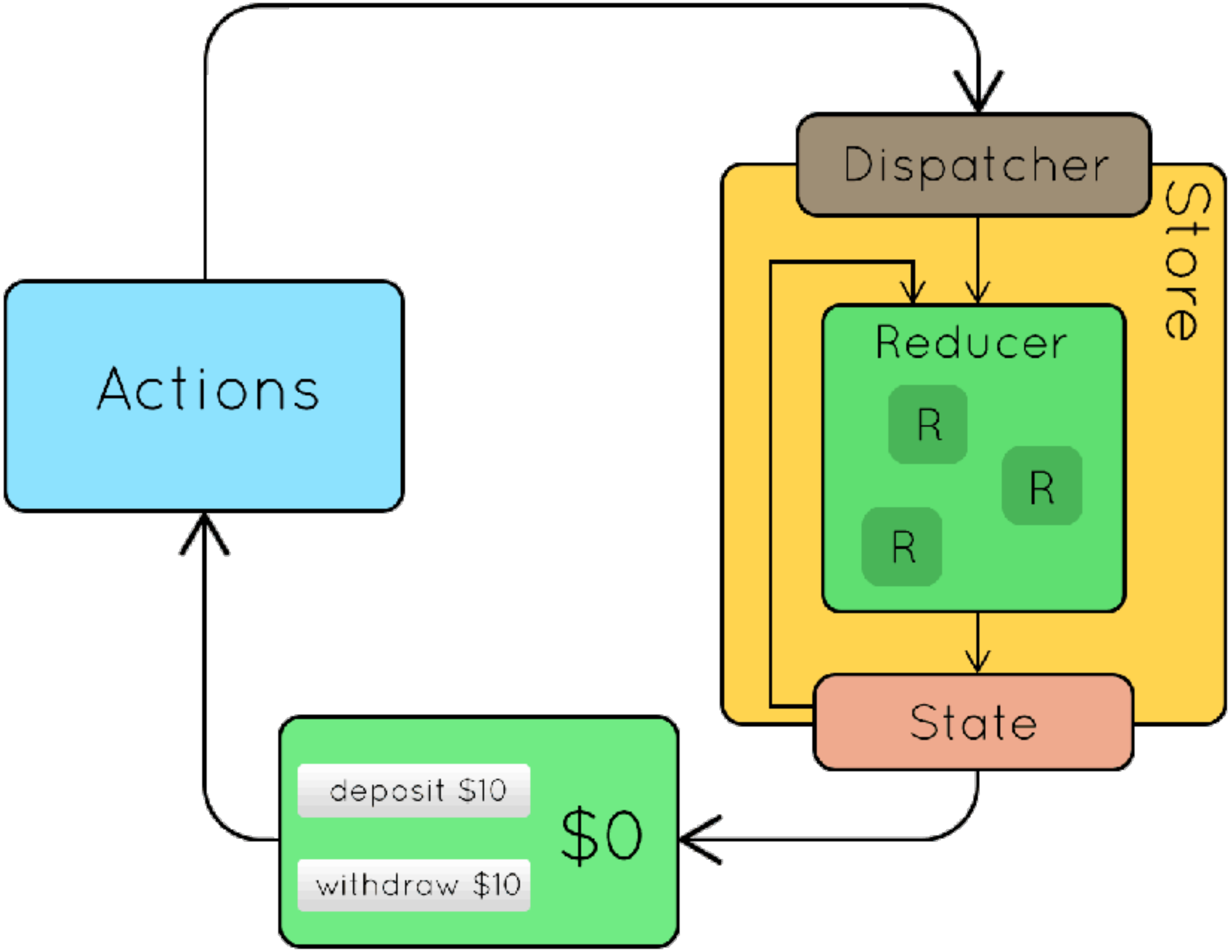
```
import { connect } from 'react-redux'

const Counter = (props) => {
    return (
        <View style={styles.counter}>
            <Text>
                {props.counter}
            </Text>
            <Button
                onPress={() => {props.dispatch({type: 'INCREMENT'})}}
                title="Increment"
            />
            <Button
                onPress={() => {props.dispatch({type: 'DECREMENT'})}}
                title="Decrement"
            />
        </View>
    );
}
```

Counter available via props

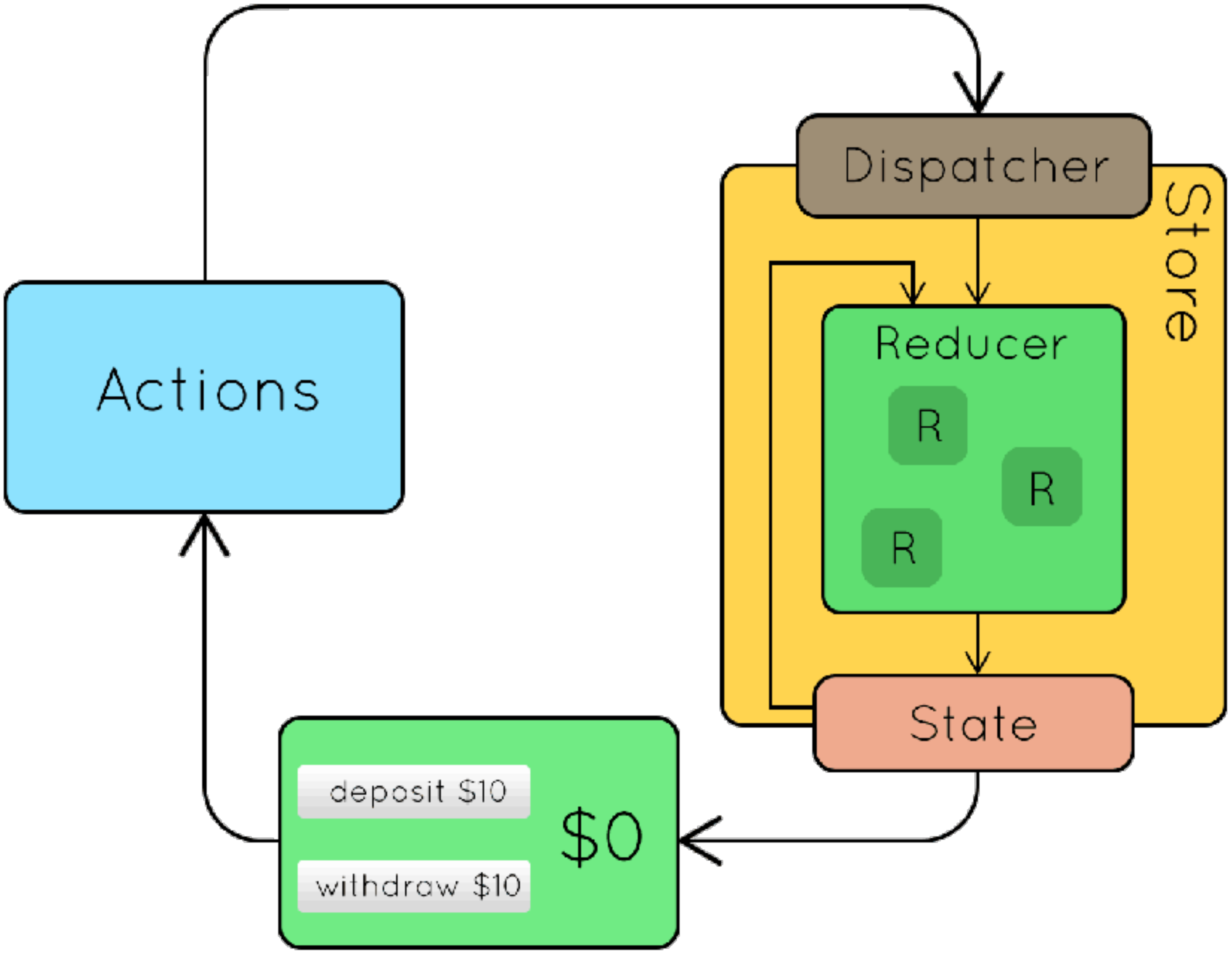
```
const mapStateToProps = (state) => {
    return {
        counter: state.counter,
    }
};

export default connect(mapStateToProps)(Counter);
```

Export the connected component

Review


Dispatcher
Store
Reducer
R
R
R
State
Actions
deposit $10
withdraw $10
$0

Review


Dispatcher
Store
Reducer
R
R
R
State
Actions
deposit $10
withdraw $10
$0

Demo

https://github.com/nacmartin/ReduxIntro

Tips

Keep your reducers pure

- Absolutely **deterministic**: the same input produces the same result every time).
- **No side effects**: no calls to an API, no changes to the outside world.

Combine reducers

```
import { combineReducers } from 'redux'
import counterReducer from './counter';
import todoReducer from './todo';

export default combineReducers({
      todo,
      counter
});
```

Model your state as a tree and write reducers for its branches

Selectors

```
const getVisibleTodos = (todos, filter) => {
      switch (filter) {
                  case 'SHOW_ALL':
                            return todos
                  case 'SHOW_COMPLETED':
                            return todos.filter(t => t.completed)
                 case 'SHOW_ACTIVE':
                            return todos.filter(t => !t.completed)
                }
}

const mapStateToProps = (state) => {
      return { todos: getVisibleTodos(state.todos, state.visibilityFilter) }
}

export default VisibleTodoList = connect(
      mapStateToProps,
)(TodoList)
```

Normalize data in the store and use selectors to derive data

Selectors with reselect

```
import { createSelector } from 'reselect'

const getVisibilityFilter = (state) => state.visibilityFilter
const getTodos = (state) => state.todos

export const getVisibleTodos = createSelector(
    [ getVisibilityFilter, getTodos ],
    (visibilityFilter, todos) => {
        switch (visibilityFilter) {
            case 'SHOW_ALL':
                return todos
            case 'SHOW_COMPLETED':
                return todos.filter(t => t.completed)
            case 'SHOW_ACTIVE':
                return todos.filter(t => !t.completed)
        }
    }
)
```

Memoized version, only computed when state.todos or state.visibilityFilter change

Use constants for action types

```
const INCREMENT_COUNTER = 'INCREMENT_COUNTER';
```

```
const increment = {
    type: INCREMENT_COUNTER
}
```

```
import { INCREMENT_COUNTER, DECREMENT } from './actionTypes'
```

Helps us to keep track of existing types, and detect typos

Use action creators

```
export function increment() {
    return {
        type: INCREMENT
    }
}

export function addTodo(text) {
    return {
        type: ADD_TODO,
        text
    }
}
```

Use action creators

```
import { increment } from ‘../actions';

const Counter = (props) => {
    return (
        <View style={styles.counter}>
            <Button
                onPress={() => {props.onIncrementClick()}}
                title="Increment" />
        </View>
    );
}
```

```
 const mapDispatchToProps = (dispatch) => {
       return {
               onIncrementClick: () => {
                         dispatch(increment())
                       }
             }
 }

 export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

ducks-modular-redux

```
const LOAD   = 'my-app/widgets/LOAD';
const CREATE = 'my-app/widgets/CREATE';
const UPDATE = 'my-app/widgets/UPDATE';
const REMOVE = 'my-app/widgets/REMOVE';

// Reducer
export default function reducer(state = {}, action = {}) {
  switch (action.type) {
    // do reducer stuff
    default: return state;
  }
}


// Action Creators
export function loadWidgets() {
  return { type: LOAD };
}

export function createWidget(widget) {
  return { type: CREATE, widget };
}

export function updateWidget(widget) {
  return { type: UPDATE, widget };
}

export function removeWidget(widget) {
  return { type: REMOVE, widget };
}
```

ducks-modular-redux

```
const LOAD   = 'my-app/widgets/LOAD';
const CREATE = 'my-app/widgets/CREATE';
const UPDATE = 'my-app/widgets/UPDATE';
const REMOVE = 'my-app/widgets/REMOVE';

// Reducer
export default function reducer(state = {}, action = {}) {
  switch (action.type) {
    // do reducer stuff
    default: return state;
  }
}


// Action Creators
export function loadWidgets() {
  return { type: LOAD };
}

export function createWidget(widget) {
  return { type: CREATE, widget };
}

export function updateWidget(widget) {
  return { type: UPDATE, widget };
}

export function removeWidget(widget) {
  return { type: REMOVE, widget };
}
```

Action types

ducks-modular-redux

```
const LOAD   = 'my-app/widgets/LOAD';
const CREATE = 'my-app/widgets/CREATE';
const UPDATE = 'my-app/widgets/UPDATE';
const REMOVE = 'my-app/widgets/REMOVE';
```

Action types

```
// Reducer
export default function reducer(state = {}, action = {}) {
  switch (action.type) {
    // do reducer stuff
    default: return state;
  }
}
```

Reducer

```
// Action Creators
export function loadWidgets() {
  return { type: LOAD };
}

export function createWidget(widget) {
  return { type: CREATE, widget };
}

export function updateWidget(widget) {
  return { type: UPDATE, widget };
}

export function removeWidget(widget) {
  return { type: REMOVE, widget };
}
```

ducks-modular-redux

```
const LOAD   = 'my-app/widgets/LOAD';
const CREATE = 'my-app/widgets/CREATE';
const UPDATE = 'my-app/widgets/UPDATE';
const REMOVE = 'my-app/widgets/REMOVE';
```

Action types

```
// Reducer
export default function reducer(state = {}, action = {}) {
  switch (action.type) {
    // do reducer stuff
    default: return state;
  }
}
```

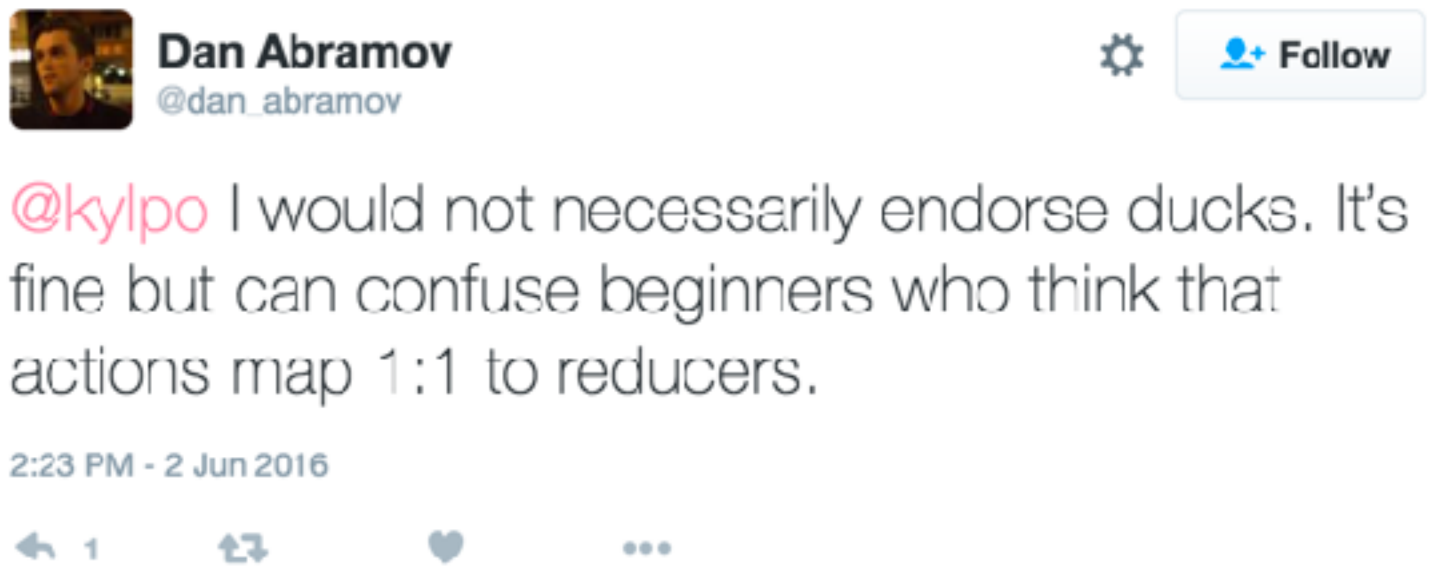
Reducer

```
// Action Creators
export function loadWidgets() {
  return { type: LOAD };
}

export function createWidget(widget) {
  return { type: CREATE, widget };
}

export function updateWidget(widget) {
  return { type: UPDATE, widget };
}

export function removeWidget(widget) {
  return { type: REMOVE, widget };
}
```

Action creators

ducks-modular-redux

Informed consent



@kylpo I would not necessarily endorse ducks. It's fine but can confuse beginners who think that actions map 1:1 to reducers.

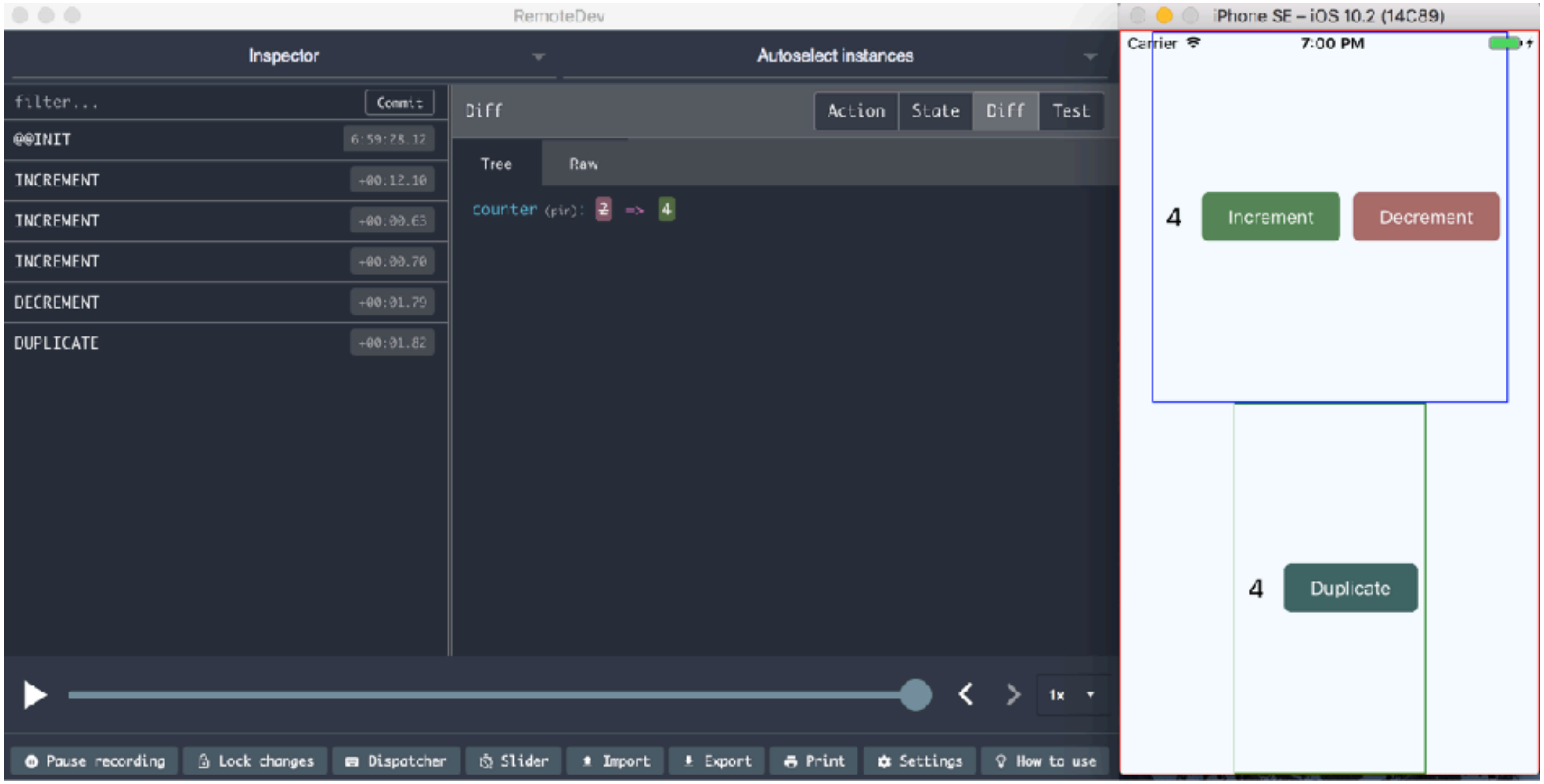
2:23 PM - 2 Jun 2016

Awesome DevTools

```
npm install --save-dev remote-redux-devtools
```

```
import devToolsEnhancer from 'remote-redux-devtools';

let store = createStore(counterReducer, devToolsEnhancer());
```

Awesome DevTools



```
@@INIT        6:59:28.12
INCREMENT     +00:12.10
INCREMENT     +00:00.63
INCREMENT     +00:00.70
DECREMENT     +00:01.79
DUPLICATE     +00:01.82
```

```
counter (pin): 2 => 4
```

Thanks!

@nacmartin

nacho@limenius.com

Limenius

simple solutions for complex needs

http://limenius.com