Final Thoughts and Code for Counter App

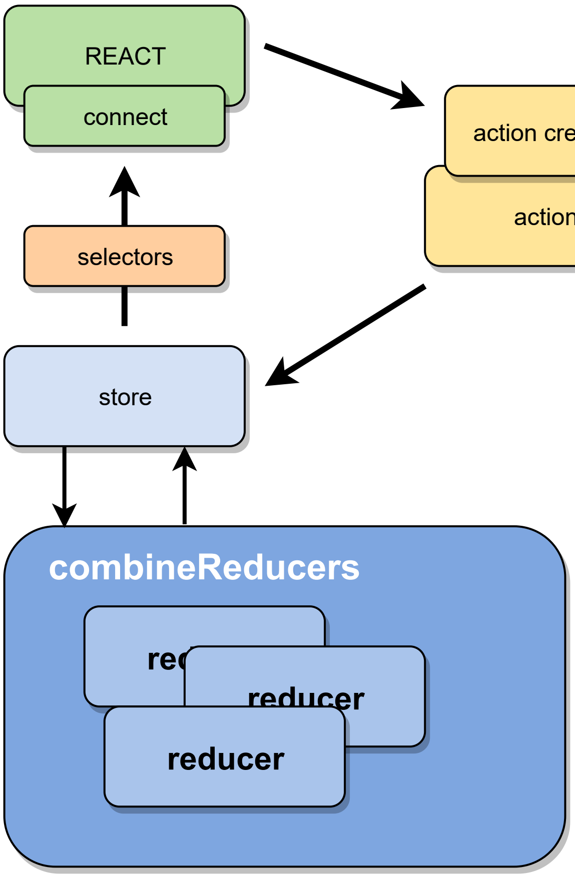
This lesson presents some final thoughts and complete, executable code for Simple Counter Application.

WE'LL COVER THE FOLLOWING

- Final thoughts
- Quick Quiz on Redux

Final thoughts

Redux is a wonderful pattern. Over the years the JavaScript community developed the idea and enhanced it with a couple of new terms. A typical redux application may look like this:



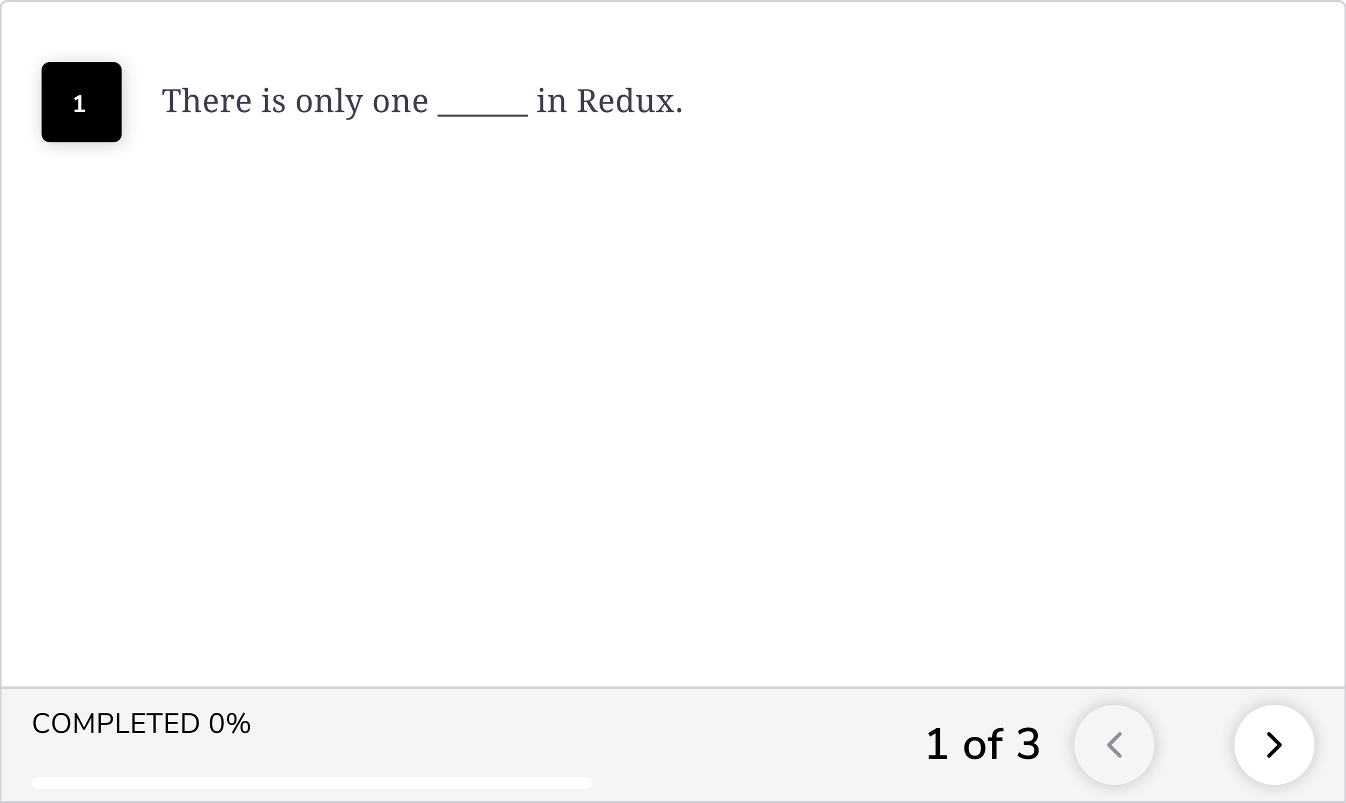
Redux Architecture

We can conclude that Redux itself is a pretty simple pattern. It teaches very useful techniques but unfortunately, it is very often not enough. Sooner or later we have to introduce more concepts/patterns. Which of course is not that bad. We just have to plan for it.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './style.css';



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

// Action creators

const ADD = 'ADD';
const SUBTRACT = 'SUBTRACT';
const CHANGE_VISIBILITY = 'CHANGE_VISIBILITY';

const add = () => ({ type: ADD });
const subtract = () => ({ type: SUBTRACT });
const changeVisibility = visible => ({ type: CHANGE_VISIBILITY, visible });

// Reducers

const initialState = {
  counter: {
    value: 0
  },
  visible: true
};
const counterReducer = function (state, action) {
  console.log(state, action);
  if (action.type === ADD) {
    return { value: state.value + 1 };
  } else if (action.type === SUBTRACT) {
    return { value: state.value - 1 };
  }
  return state || { value: 0 };
};
const visibilityReducer = function (state, action) {
  if (action.type === CHANGE_VISIBILITY) {
    return action.visible;
  }
  return true;
};
const rootReducer = combineReducers({
  counter: counterReducer,
  visible: visibilityReducer
});

// Selectors

const getCounterValue = state => state.counter.value;
const getVisibility = state => state.visible;

// Store creation
```

```
const store = createStore(rootReducer, initialState);


// React components

function Counter({ value, add, subtract }) {
  return (
    <div>
      <p>Value: { value }</p>
      <button onClick={ add }>Add</button>
      <button onClick={ subtract }>Subtract</button>
    </div>
  );
}
const CounterConnected = connect(
  state => ({
    value: getCounterValue(state)
  }),
  dispatch => ({
    add: () => dispatch(add()),
    subtract: () => dispatch(subtract())
  })
)(Counter);

function Visibility({ changeVisibility }) {
  return (
    <div>
      <button onClick={ () => changeVisibility(true) }>Visible</button>
      <button onClick={ () => changeVisibility(false) }>Hidden</button>
    </div>
  );
}
const VisibilityConnected = connect(
  null,
  dispatch => ({
    changeVisibility: value => dispatch(changeVisibility(value))
  })
)(Visibility);



function App({ visible }) {
  return (
    <div>
      <VisibilityConnected />
      { visible && <CounterConnected /> }
    </div>
  );
}

const AppConnected = connect(
  state => ({
    visible: getVisibility(state)
  })
)(App);

ReactDOM.render(<Provider store={ store }><AppConnected /></Provider>, document.getElementByI
```

Quick Quiz on Redux



1 There is only one ______ in Redux.

This is about it on Data Flow in React. There are a few other important topics that we are going to discuss in the next chapter.