Chapter 29: React with Redux

Redux has come to be the status quo for managing application-level state on the front-end these days, and those who work on "large-scale applications" often swear by it. This topic covers why and how you should use the state management library, Redux, in your React applications.

Section 29.1: Using Connect

Create a Redux store with *createStore*.

```
import { createStore } from 'redux'
import todoApp from './reducers'
let store = createStore(todoApp, { inistialStateVariable: "derp"})
```

Use *connect* to connect component to Redux store and pull props from store to component.

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

const VisibleTodoList = connect(
  mapStateToProps,
  mapDispatchToProps
)(TodoList)

export default VisibleTodoList
```

Define actions that allow your components to send messages to the Redux store.

```
/*
 * action types
 */

export const ADD_TODO = 'ADD_TODO'

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

Handle these messages and create a new state for the store in reducer functions.

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