COMP 431/531: Web Development

Lecture 11: React Hooks and Redux

Mack Joyner (mjoyner@rice.edu)

https://www.clear.rice.edu/comp431

In-Class Exercise 10: React Tic-Tac-Toe

Announcements & Reminders

- Quiz #2 (Events, Storage, Arrays) due today at 11:59pm
- HW #4 (Draft Front-end) due Tue. Oct. 21st at 11:59pm

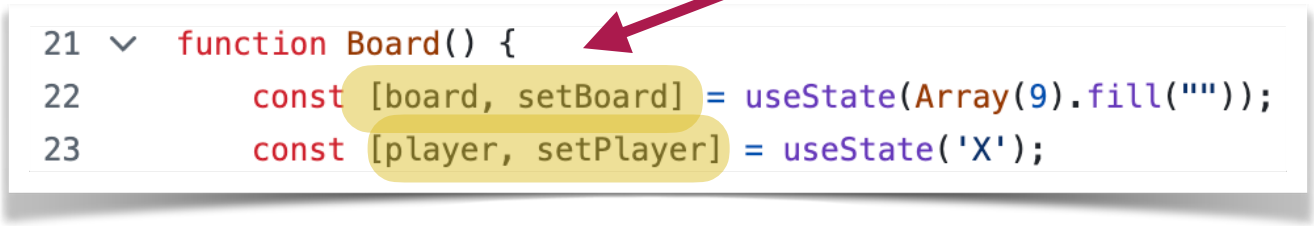
classroom hw4 repo: https://classroom.github.com/a/hfLD_40C

Add React to Chrome Dev Tools

Chrome Web Store: Home -> Extensions -> React Development Tools

Functions with State (Local)

State held in functions (e.g. Board)



```
21  function Board() {
22      const [board, setBoard] = useState(Array(9).fill(""));
23      const [player, setPlayer] = useState('X');
```


Functions with State (Local)

```
30      function handleClick(i) {
31          // TODO update board state
32          if (!board[i]) {
33              setBoard(() => {
34                  let newBoard = [...board];
35                  newBoard[i] = player;
36                  return newBoard
37              });
38              setPlayer(() => player == 'X' ? 'O' : 'X');
39          }
40      }
```

Board nested function modifies state

Functions with State (Local)

Board passes state as prop to Square

```
26          function renderSquare(i) {
27              return <Square value={board[i]} onClick={() => handleClick(i)}/>;
28          }
```

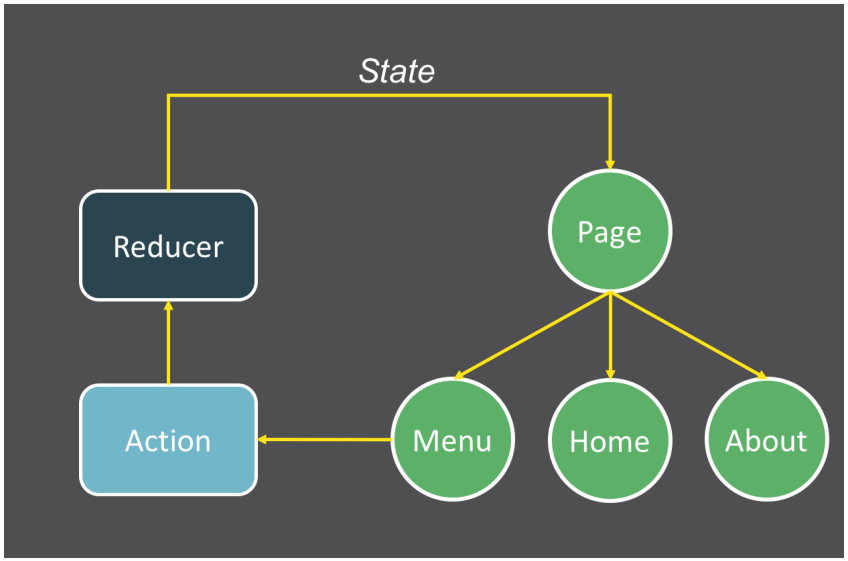

Idea: Lean Components

- Components are presentational
- Components have very little if any state
- Components dispatch actions to update "global" state
- The "global" state is available to all Components

Redux

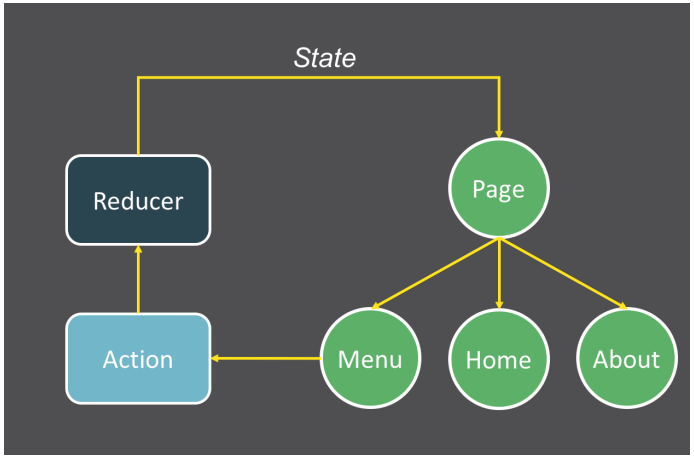
JavaScript library for managing app state

>> npm install --save @reduxjs/toolkit react-redux



Redux Principles

- Redux manages single immutable state store
  - Acts as central location to access global state of whole app
- State changes occur by dispatching actions
  - Component unaware of how state changes
  - Dispatch reducer (function) and return new state (no mutation)
- Reducer: takes state, action => next state
  - Function has to be pure (no mutation of current state)
- Easier to write reducer, reducer test

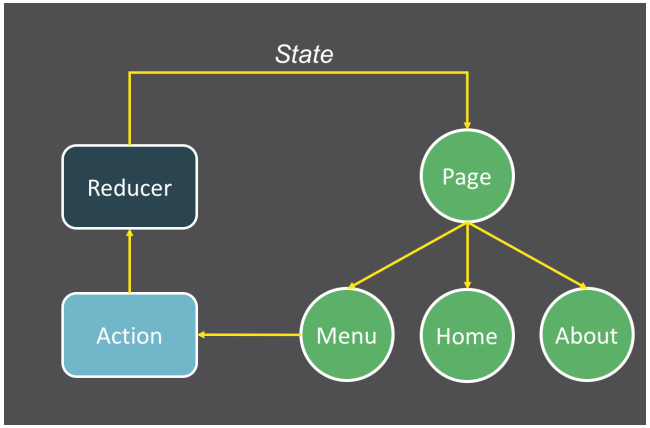


Roadmap

1. Global store (Redux reducer)
   - State
   - Actions
2. Connect React VDOM to Redux store
3. Reducer
   - Used to update global store
   - Takes as input current state and action
   - Returns updated state
4. React example that uses Redux store

Redux - Global Store (tictactoeSlice.tsx)

- Reducer takes a state, action and returns a state
- No mutation, old state is not modified
- Components dispatch actions to update state
- Define initial state, state changes by action
- Returned state is now saved as global state



State (tictactoeSlice.tsx)

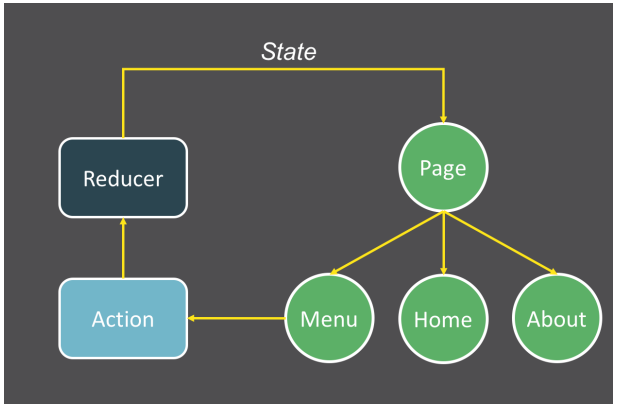
```tsx
import { createSlice } from '@reduxjs/toolkit';

export const tictactoeSlice = createSlice({
    name: 'game',
    initialState: {
        players:[],
        nextPlayer: 'X',
        series: {xWins: 0, oWins: 0},
        status: 'Player turn: ',
        board: Array(9).fill("")
    },
    reducers: {
        selectSquare: (state, action) => {
            const id = 0; //TODO fill in
            if (!state.board[id]) {
                state.board[id] = state.nextPlayer;
            }
        }
    }
})

// Generate action creators
export const { selectSquare } = tictactoeSlice.actions

export default tictactoeSlice.reducer
```

Redux state slice

Initial state

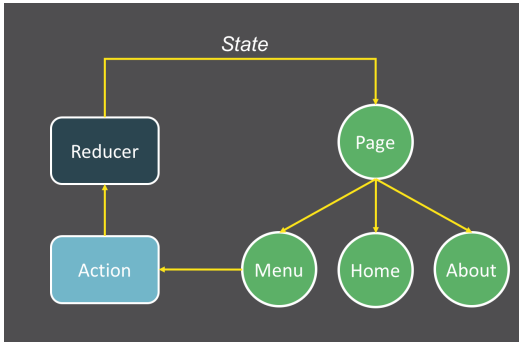
Functions defined to update state

Reducer with global state, actions



Connect to Redux store (main.tsx)

- Connect redux store to React VDOM
- Wrap <Provider> around main component
- All children components may now use reducer




```
1   import { StrictMode } from 'react'
2   import { createRoot } from 'react-dom/client'
3   import { Provider } from 'react-redux';
4   import { configureStore } from '@reduxjs/toolkit';
5   import tictactoeReducer from "./features/game/tictactoeSlice";
6   import './index.css'
7   import App from './App.tsx'
8
9   const store = configureStore({reducer: {game: tictactoeReducer }})
10
11  createRoot(document.getElementById('root')!).render(
12      <StrictMode>
13          <Provider store={store}>
14              <App />
15          </Provider>
16      </StrictMode>
17  )
```

Define store with reducer

Wrap store around App

Presentational Board using Redux (Board.tsx)

- Import redux hooks
  - useSelector (state)
  - useDispatch (dispatch actions)
- Board locally declare var state
  - potentially updates after action
  - dispatch reducer functions
- Can pass action object to reducer function
  - Optionally only pass primitive variable
  - Reducer pulls data from action payload

```
2    import { useSelector, useDispatch } from 'react-redux';
3    import { selectSquare } from './features/game/tictactoeSlice';
4  > const Square = ({value, selectSquare}) => ( ...
8            );
9
10 v function Board() {
11       const tttBoard = useSelector((state) => state.game.board)

15       const dispatch = useDispatch();

42 v     function renderSquare(i) {
43           return (
44               <Square value={tttBoard[i]}
45                       selectSquare={() => dispatch(selectSquare({id: i}))}
46               />);
```

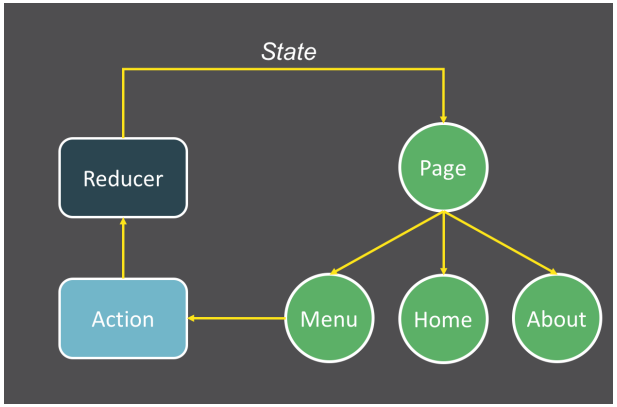
React hook to bind state

React hook to dispatch actions

Dispatch reducer function with action

State (tictactoeSlice.tsx)

```
1   import { createSlice } from '@reduxjs/toolkit';
2
3   export const tictactoeSlice = createSlice({
4       name: 'game',
5       initialState: {
6           players:[],
7           nextPlayer: 'X',
8           series: {xWins: 0, oWins: 0},
9           status: 'Player turn: ',
10          board: Array(9).fill("")
11      },
12      reducers: {
13          selectSquare: (state, action) => {
14              const id = 0; //TODO fill in
15              if (!state.board[id]) {
16                  state.board[id] = state.nextPlayer;
17              }
18          }
19      }
20  })
21
22  // Generate action creators
23  export const { selectSquare } = tictactoeSlice.actions
24
25  export default tictactoeSlice.reducer
```



Access action object by using action.payload

Using Immer, spread op (…) not needed

Updates and returns new global state, no mutation

Presentational Board using Redux (Board.tsx)

- Import redux hooks
  - useSelector (state)
  - useDispatch (dispatch actions)
- Board locally declare var state
  - potentially updates after action
  - dispatch reducer functions
- Can pass action object to reducer function
  - Optionally only pass primitive variable
  - Reducer pulls data from action payload

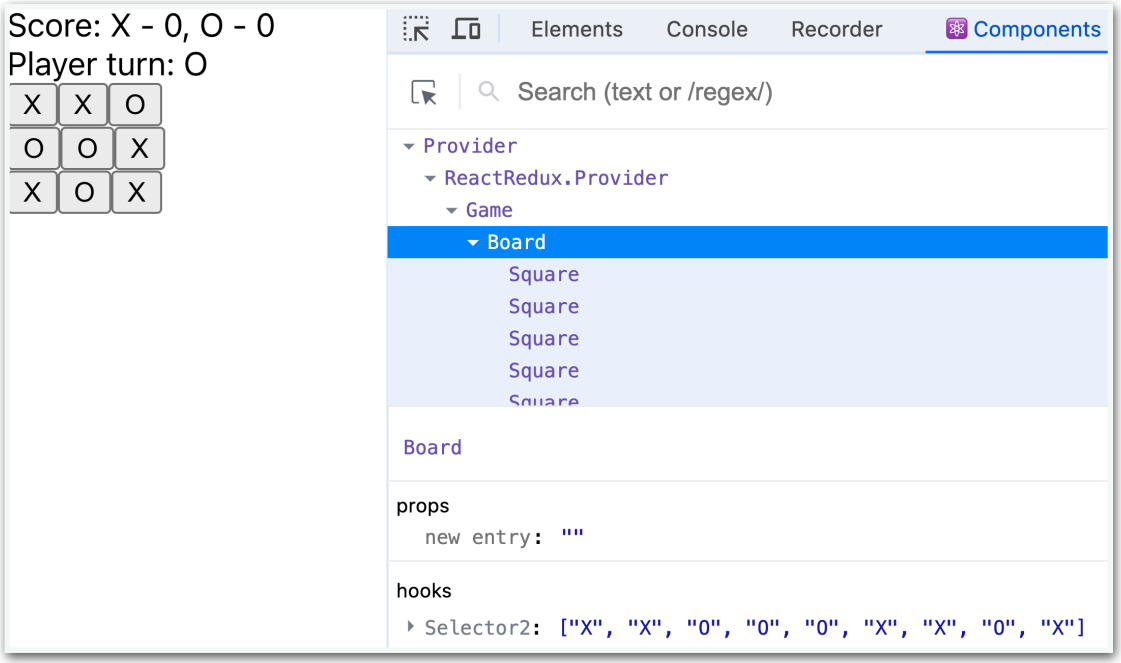
```
2    import { useSelector, useDispatch } from 'react-redux';
3    import { selectSquare } from './features/game/tictactoeSlice';
4    const Square = ({value, selectSquare}) => ( ...
8            );
9
10   function Board() {
11       const tttBoard = useSelector((state) => state.game.board)

15       const dispatch = useDispatch();

42       function renderSquare(i) {
43           return (
44               <Square value={tttBoard[i]}
45                       selectSquare={() => dispatch(selectSquare({id: i}))}
46               />);
```

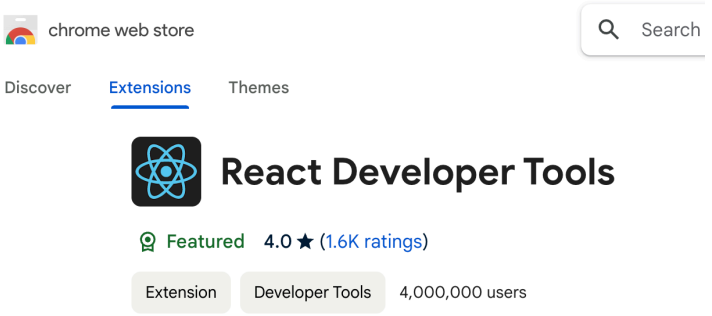
Re-render Square since state changed

Roadmap

1. Global store (Redux reducer)
   - State
   - Actions
2. Connect React VDOM to Redux store
3. Reducer
   - Used to update global store
   - Takes as input current state and action
   - Returns updated state
4. React example that uses Redux store

React Developer Tools (Redux)





Resource

https://react-redux.js.org/introduction/getting-started