

Better React state management with Redux

Maurice de Beijer

@mauricedb

@react_tutorial

Workshop goal

- Short recap: what is React?
- Learn the three principals of Redux
- Use the different Redux components
- Unit testing Redux code
- Adding Redux to a React application
- Use Redux middleware



- Maurice de Beijer
- The Problem Solver
- Microsoft Azure MVP
- Freelance developer/instructor
- Twitter: @mauricedb and @React_Tutorial
- Web: <http://www.TheProblemSolver.nl>
- E-mail: maurice.de.beijer@gmail.com



Follow along

```
1  import React, { Component, PropTypes } from 'react';  
2  
3  class NewComponent extends Component {  
4    render() {  
5      return (  
6        <div>  
7          {this.constructor.name}  
8        </div>  
    )  
  }  
}
```



- Git repository: <https://github.com/mauricedb/nitflex-redux>
- Slides: <http://bit.ly/react-redux-workshop>

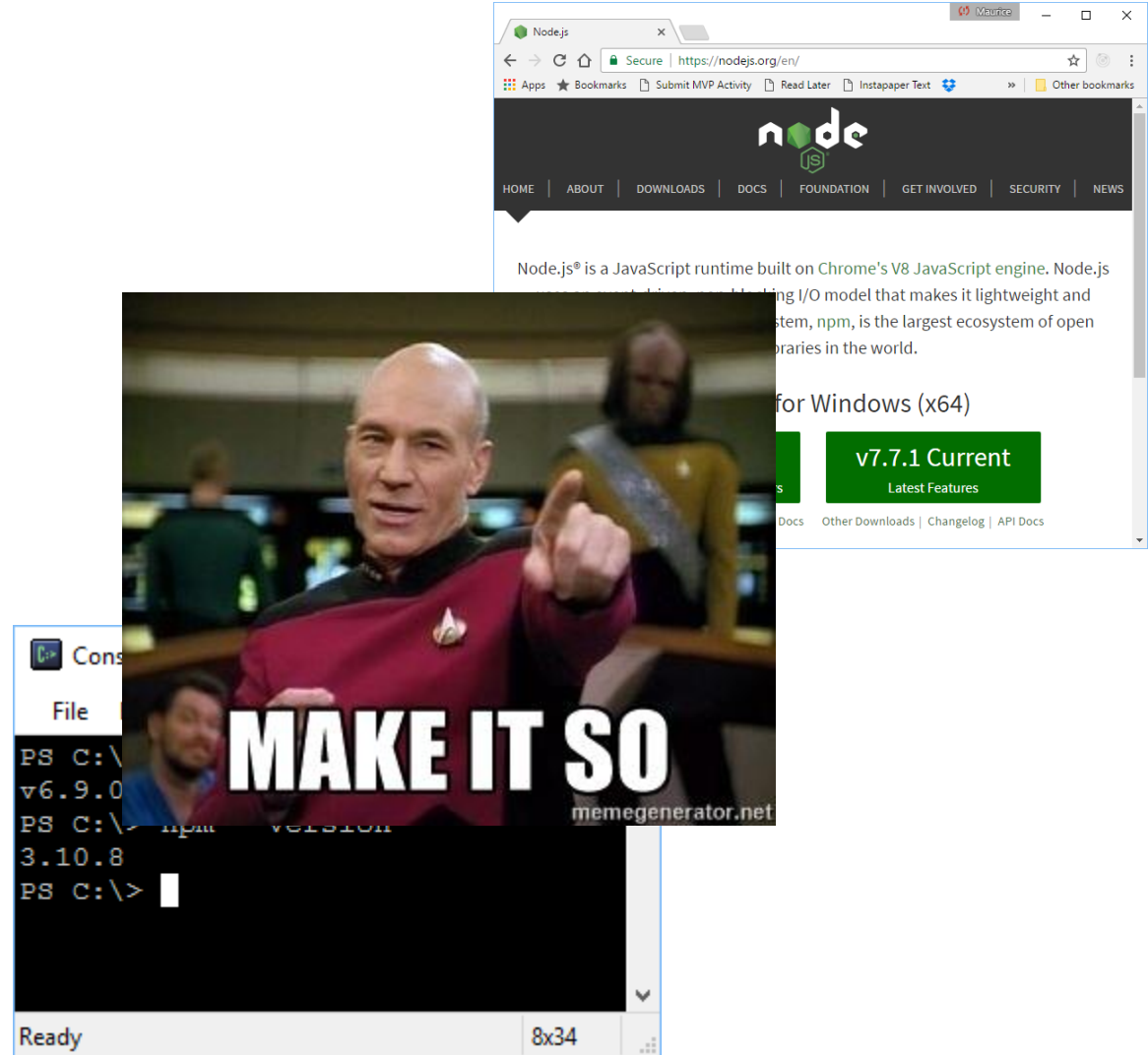
Type it out
by hand?

“Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!”

Prerequisites

Install Node & NPM

Install Node.js & NPM



Short recap

What is React?

React

“React is a JavaScript library for building user interfaces”

- Facebook -

React features

- React uses a component based architecture
- Components are written using the JSX syntax
- React likes a one way data flow
- React uses a virtual DOM

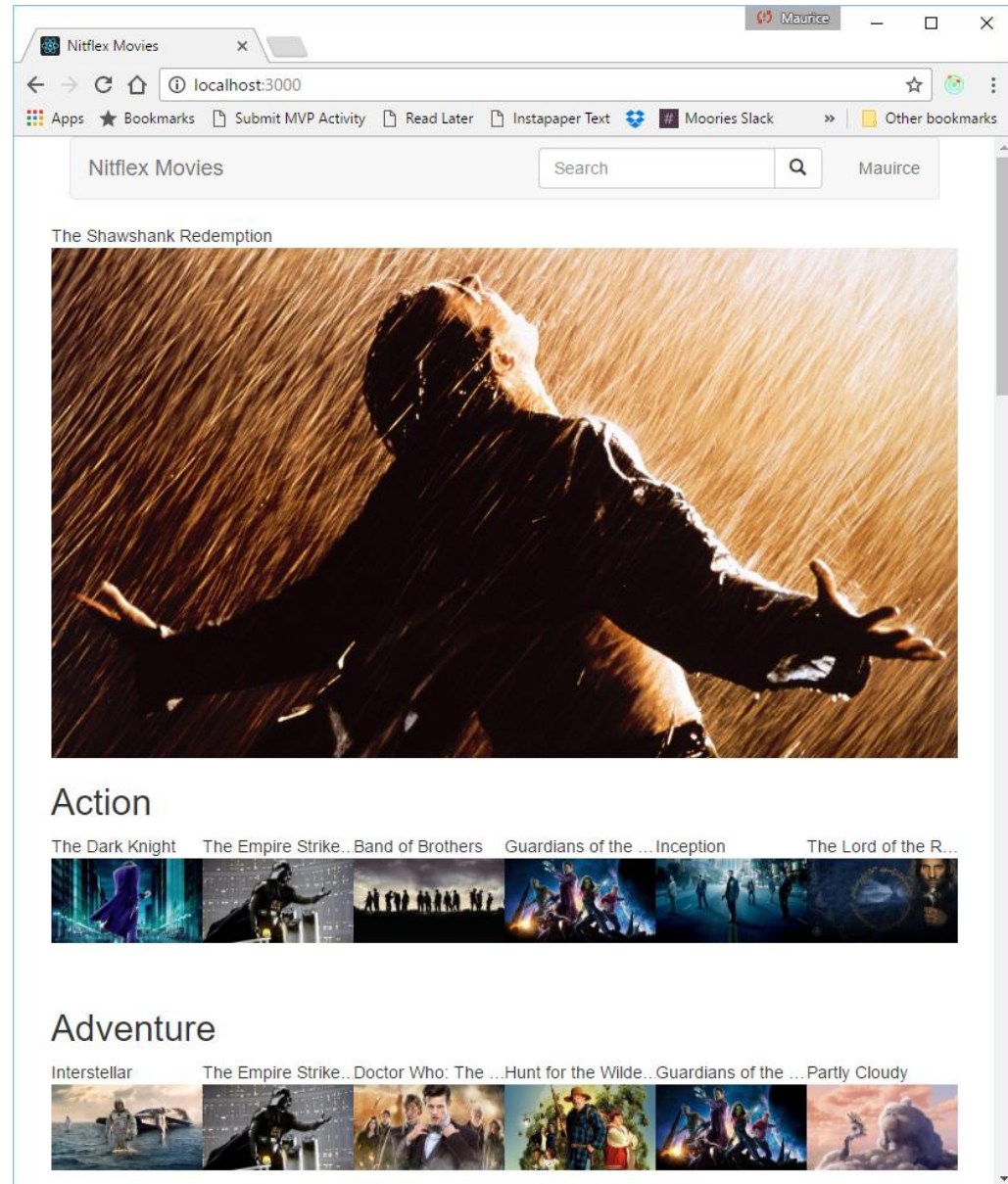
React Component

```
1  import React, { Component, PropTypes } from 'react';
2
3  class GenreRowMovie extends Component {
4    constructor(props) {
5      super(props);
6
7      this.expandMovie = this.expandMovie.bind(this);
8    }
9
10   expandMovie() {
11     this.props.expandMovie(this.props.movie);
12   }
13
14   render() {
15     const { movie } = this.props;
16
17     return (
18       <div className="col-sm-2 genre-row-movie">
19         <div className="title">
20           {movie.title}
21         </div>
22         <img
23           className="img-responsive"
24           alt={movie.title}
25           src={`http://image.tmdb.org/t/p/w300/${movie.backdrop_path}`}
26         />
27         <button className="btn btn-link btn-block expand" onClick={this.expandMovie}>
28           <i className="glyphicon glyphicon-chevron-down" />
29         </button>
30       </div>
31     );
32   }
33 }
```

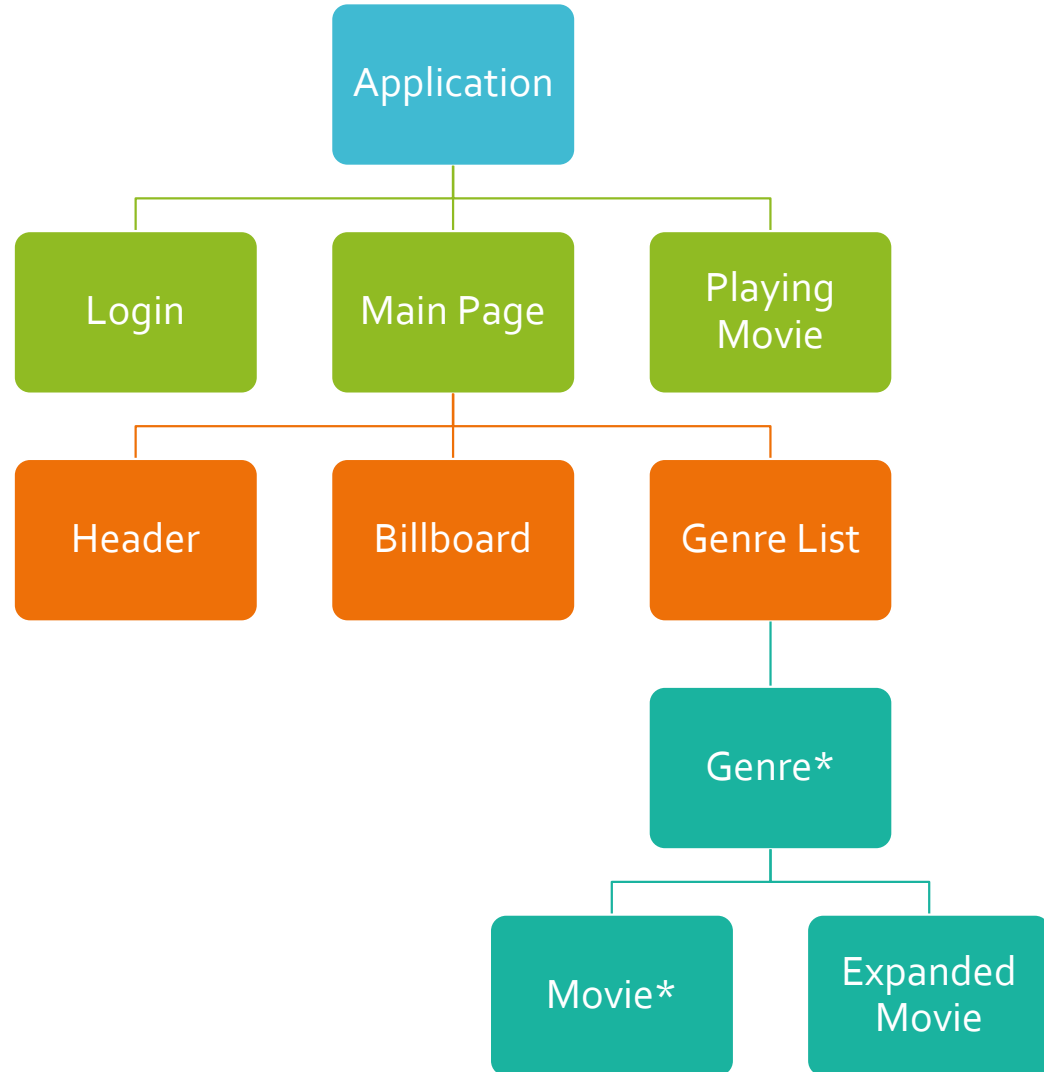
Nitflex

The sample application

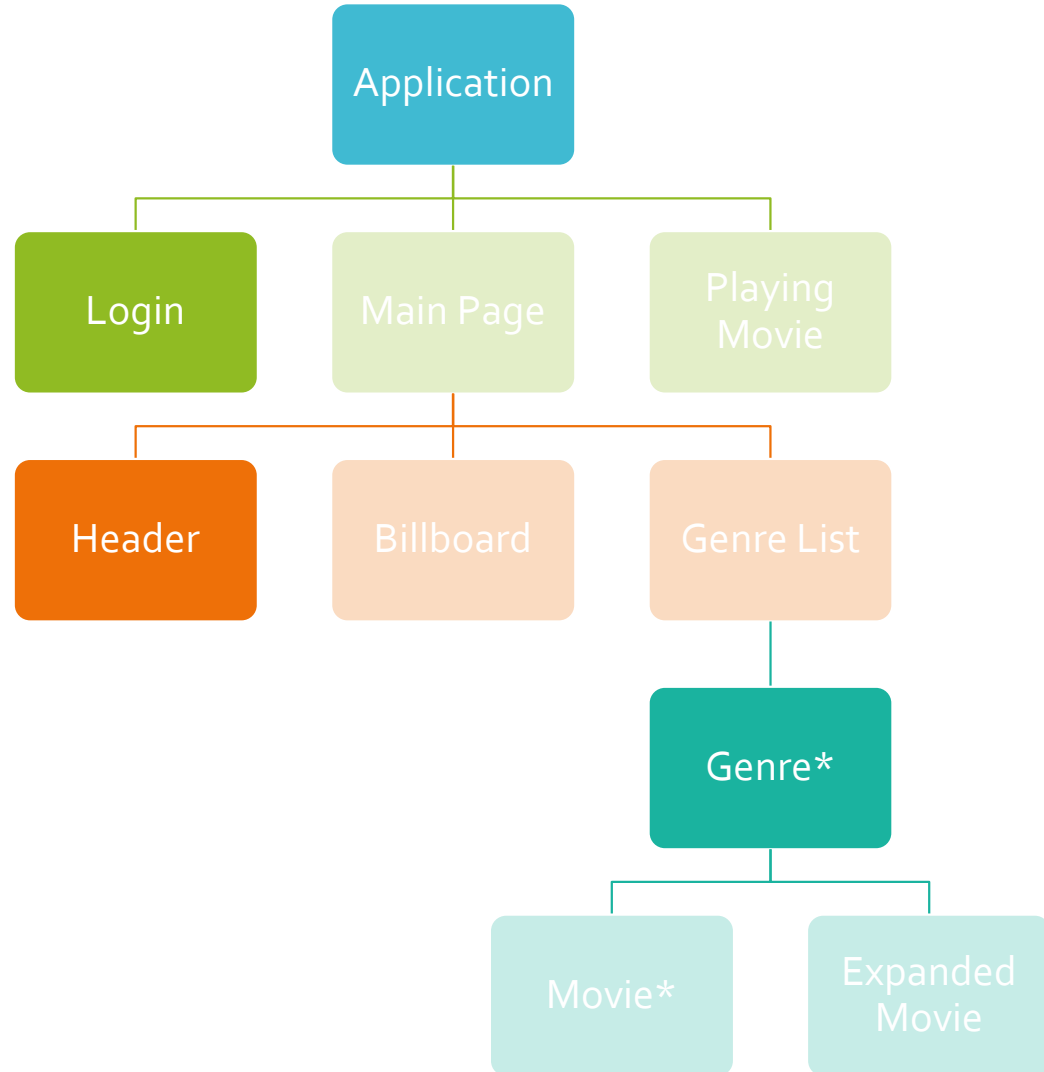
Nitflex



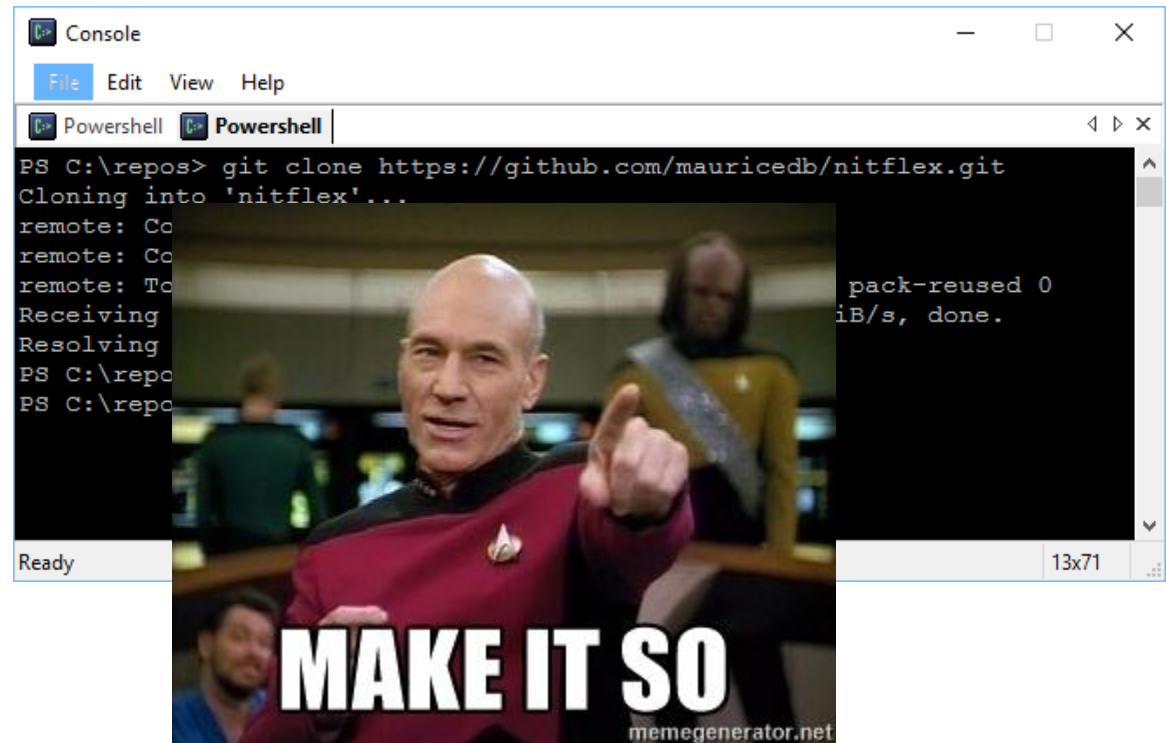
Main components



Stateful components



Clone Nitflex & NPM install



Redux

Redux

“Redux is a predictable state container for JavaScript apps”

- Dan Abramov -

Three principals of Redux

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

Single source of truth

- The application state is stored in a single location
- Do not scatter the state over lots of components
- A single state location makes it easier to reason about and debug your application

State is read-only

- The state objects are never changed by the application
- Every state change is the result of an action being dispatched to the store
- This action describes the intended state change

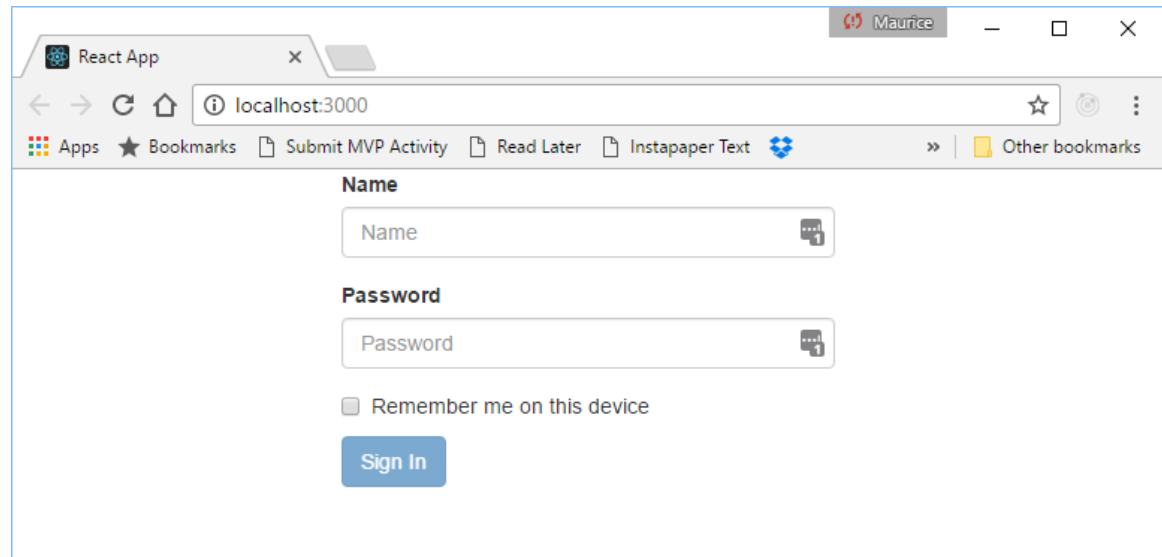
Changes with pure functions

- Reducers are pure functions that take the current state and an action as input and return a new state object
- The previous state object is never modified

UI versus App state

- Redux should be used for application state.
 - Do not store UI state in Redux!

Login example



A screenshot of a web browser window displaying a login form. The browser's address bar shows 'localhost:3000'. The page title is 'React App'. The form contains the following elements:

- Name**: A text input field with a placeholder 'Name' and a small icon of two speech bubbles with a plus sign.
- Password**: A text input field with a placeholder 'Password' and a small icon of two speech bubbles with a plus sign.
- ☐ Remember me on this device
- Sign In**: A blue button with white text.

Installing Redux



Redux Components

Redux Components

- Actions
- Reducers
- The store

Login action

```
1  export const login = user => ({  
2    type: 'LOGIN',  
3    payload: user,  
4  });  
5
```

Login reducer

```
1  const user = (state = null, action) => {  
2    switch (action.type) {  
3      case "LOGIN":  
4        if (action.payload.rememberMe) {  
5          localStorage.user = JSON.stringify(action.payload);  
6        }  
7        
8      return Object.assign({}, state, action.payload);  
9      default:  
10       return state;  
11    }  
12  };  
13  
14  export default user;
```

combineReducers

```
1  import { combineReducers } from "redux";  
2  import user from "../user";  
3  
4  const reducers = combineReducers({
```



Unit testing

Unit test the action creator

```
1  import { login } from ".";
2
3  describe("Actions", () => {
4    it("should create a LOGIN action", () => {
5      const action = login({ name: "Maurice" });
6
7      expect(action).toEqual({
8        type: "LOGIN",
9        payload: {
10          name: "Maurice"
11        }
12      });
13    });
14  });
```


Unit test the reducer

```
1  import reducer from "../user";
2  import { login } from "../actions";
3
4  describe("The user reducer", () => {
5    it("should return the initial state", () => {
6      expect(reducer(undefined, {})).toBeNull();
7    });
8
9    it
10
11
12
13
14
15
16
17   })
18   });
```

name: "Maurice" }));



Redux and React

Redux and React

- The NPM react-redux package contains the React bindings for Redux
- The <Provider> component makes the Redux store available
- The connect() function connects React components to the Redux store use in <Provider>

Create store

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  import { createStore } from 'redux';
5  import { Provider } from 'react-redux';
6
7  import AppContainer from './app-container';
8
9  import reducers from './reducers';
10
11  // npm install bootstrap --save
12  import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
13
14  const store = createStore(reducers);
15
16  ReactDOM.render(
17    <Provider store={store}>
18      <AppContainer />
19    </Provider>,
20    document.getElementById('root')
21  );
```

Dispatch login action

```
92
93 import { connect } from 'react-redux';
94 import { login } from '../actions';
95
96 const mapDispatchToProps = (dispatch) => {
97   return {
98     loginAsUser: (user) => {
99       dispatch(login(user));
100     },
101   };
102 };
103
104 export default connect(
105   null,
106   mapDispatchToProps,
107 )(LoginPage);
108
```

Get the user from the store

```
37 import { connect } from 'react-redux';
38
39 const mapStateToProps = (state, props) => Object.assign({},
40   props, {
41     user: state.user,
42   })
43
44 export default connect(
45   mapStateToProps,
46 )(AppPresentation);
47
48
```

Cleanup

- app-container.jsx

- app-



Redux Middleware

Redux Middleware

“Redux middleware provides a third-party extension point between dispatching an action, and the moment it reaches the reducer”

- Dan Abramov -

Installing redux-thunk



Redux-Thunk

What is a Thunk?

“A thunk is a subroutine that is created, often automatically, to assist a call to another subroutine”

- Wikipedia -

Configure Thunk Middleware

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  import { createStore, applyMiddleware } from 'redux';
5  import { Provider } from 'react-redux';
6  import thunk from 'redux-thunk';
7
8  import AppContainer from './app-container';
9
10 import reducers from './reducers';
11
12 // npm install bootstrap --save
13 import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
14
15 const store = createStore(reducers, applyMiddleware(thunk));
16
17 ReactDOM.render(
18   <Provider store={store}>
19     <AppContainer />
20   </Provider>,
21   document.getElementById('root')
22 );
```

Update the Actions

```
1  export const moviesLoaded = movies => ({
2    type: 'MOVIES-LOADED',
3    payload: movies,
4  });
5
6  export const loadMovies = () => dispatch =>
7    fetch('/movies.json')
8      .then(rsp => rsp.json())
9      .then(movies => dispatch(moviesLoaded(movies)));
10
11 export const login = user => (dispatch) => {
12   dispatch({
13     type: 'LOGIN',
14     payload: user,
15   });
16   dispatch(loadMovies());
17 }
```

Add movies reducer

```
1  const movies = (state = {}, action) => {  
2    switch (action.type) {  
3      case "MOVIES-LOADED":  
4        return Object.assign({}, state, { movies: [...action.payload] });  
5        
6      default:  
7        return state;  
8    }  
9  };  
10  
11  export default movies;
```

Update the reducers index.js

```
1  import { combineReducers } from "redux";
2  import movies from "../movies";
3  import user from "../user";
4
5  const reducers = combineReducers({
6    movies,
7    user
8  });
9
10 export default reducers;
```


Extract movies from store

```
37   const mapStateToProps = (state, props) => Object.assign({},  
38     props, {  
39       user: state.user,  
40       movies: state.movies.movies,  
41     })  
42  
43   export default connect(  
44     mapStateToProps,  
45   })(AppPresentation);  
46
```

Cleanup

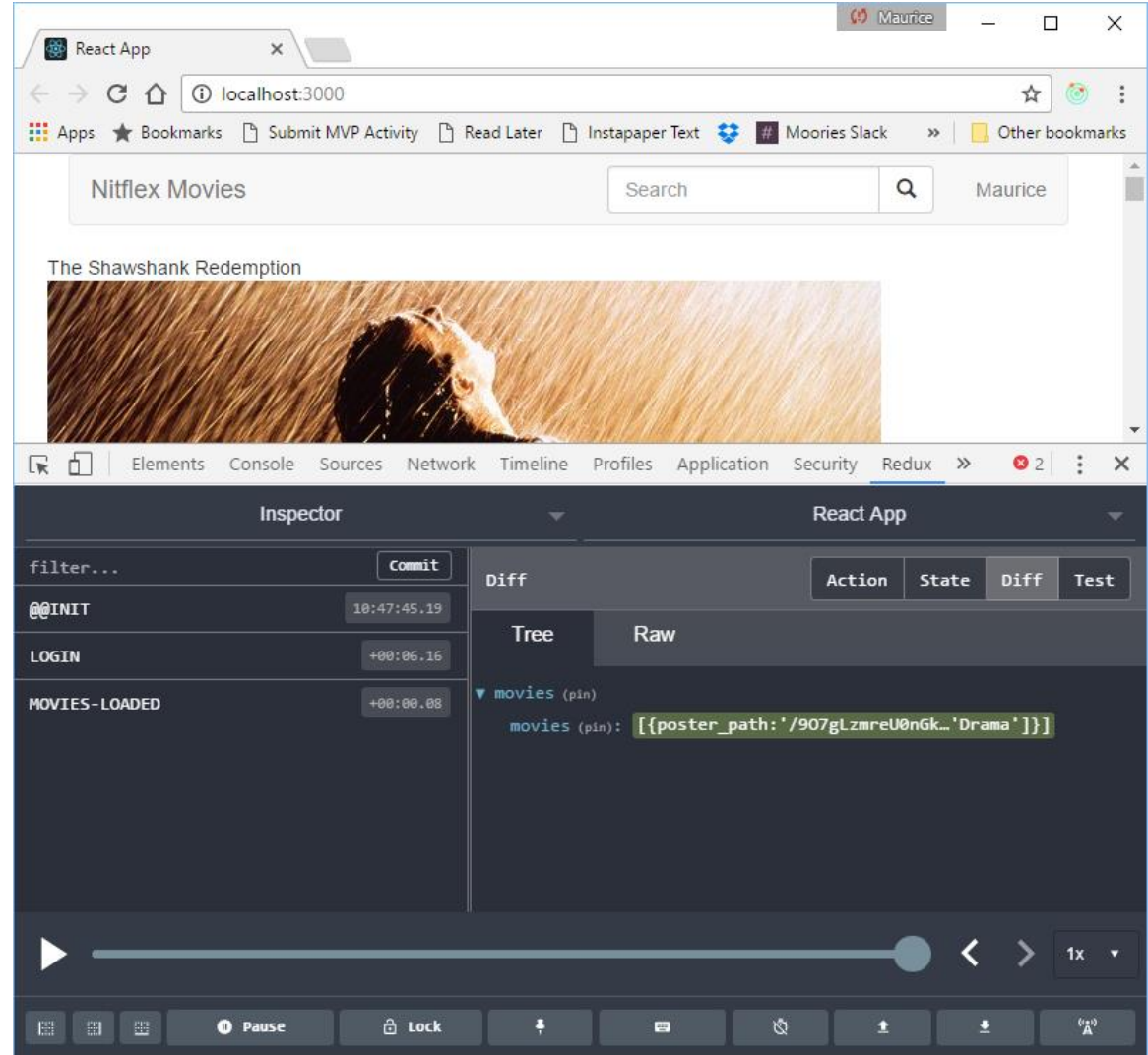
- app-container-icx
-

ies to



Redux DevTools

Redux DevTools



Redux DevTools

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3
4  import { createStore, applyMiddleware, compose } from "redux";
5  import { Provider } from "react-redux";
6  import thunk from "redux-thunk";
7
8  import AppContainer from "../app-container";
9
10 import reducers from "../reducers";
11
12 // npm
13 import
14
15 const
16 const
17
18 ReactDOM
19 <Pro
20 <A
21 </Pro
22 docu
23 );
```

css";

N_COMPOSE__ || compose;
yMiddleware(thunk)))





Remember the user

Remember
the user

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3
4  import { createStore, applyMiddleware } from "redux";
5  import { Provider } from "react-redux";
6  import thunk from "redux-thunk";
7
8  import AppContainer from "../app-container";
9
10 import reducers from "../reducers";
11 import { login } from "../actions";
12
13 // npm install bootstrap --save
14 import "../node_modules/bootstrap/dist/css/bootstrap.min.css";
15
16 are(thunk));
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32 document.getElementById("root")
33 );
```



memegenerator.net

Playing and stopping movies

Add actions

```
1  export const moviesLoaded = movies => ({
2    type: "MOVIES-LOADED",
3    payload: movies
4  });
5
6  export const loadMovies = () =>
7    dispatch =>
8      fetch("/movies.json")
9        .then(rsp => rsp.json())
10       .then(movies => dispatch(moviesLoaded(movies)));
11
12  export const login = user => dispatch => {
13    dispatch({
14      type: "LOGIN",
15      payload: user
16    });
17    dispatch(loadMovies());
18  };
19
20  export const startPlayingMovie = movie => ({
21    type: 'START-PLAYING',
22    payload: movie,
23  });
24
25  export const stopPlayingMovie = () => ({
26    type: 'STOP-PLAYING',
27  });
28
29
```

Update reducer

```
1  const movies = (state = {}, action) => {
2    switch (action.type) {
3      case "MOVIES-LOADED":
4        return Object.assign({}, state, { movies: [...action.payload] });
5
6      case 'START-PLAYING':
7        return Object.assign({}, state, { playing: action.payload });
8
9      case 'STOP-PLAYING':
10       return Object.assign({}, state, { playing: null });
11
12     default:
13       return state;
14   }
15 };
16
17 export default movies;
18
```

Start playing

```
49
50 import { connect } from 'react-redux';
51 import { startPlayingMovie } from '../actions';
52
53 const mapDispatchToProps = dispatch => ({
54   startPlaying: (movie) => {
55     dispatch(startPlayingMovie(movie));
56   },
57 });
58
59 export default connect(
60   null,
61   mapDispatchToProps,
62 )(ExpandedGenreRowMovie);
```

Retrieve the
playing
movie from
the store in
app-
presentation

```
43  const mapStateToProps = (state, props) => Object.assign({}, props, {  
44    user: state.user,  
45    movies: state.movies.movies,  
46    playing: state.movies.playing  
47  });  
48  
49  export default connect(mapStateToProps)(AppPresentation);
```

Retrieve the
playing
movie from
the store in
playing-
movie

```
30   const mapStateToProps = (state) => ({  
31     movie: state.movies.playing  
32   });  
33  
34   const mapDispatchToProps = dispatch => ({  
35     stopPlaying: movie => dispatch(stopPlayingMovie(movie)),  
36   });  
37  
38   export default connect(  
39     mapStateToProps,  
40     mapDispatchToProps,  
41   })(PlayingMovie);  
42
```

Cleanup

- app-container.jsx
- main
- genre
- genre



Filtering movies

Filtering movies

- The list of filtered movies are derived from state

Add action

```
28  
29 export const filterMovies = search => ({  
30   type: 'FILTER-MOVIES',  
31   payload: search,  
32 });  
33
```

Reducer

```
1  const movies = (state = {}, action) => {
2    switch (action.type) {
3      case "MOVIES-LOADED":
4        return Object.assign({}, state, { movies: [...action.payload] });
5
6      case 'START-PLAYING':
7        return Object.assign({}, state, { playing: action.payload });
8
9      case 'STOP-PLAYING':
10       return Object.assign({}, state, { playing: null });
11
12     case 'FILTER-MOVIES':
13       return Object.assign({}, state, { searchText: action.payload });
14
15     default:
16       return state;
17   }
18 };
19
20 export default movies;
```

Filter- movies.jsx

```
51  FilterMovies.propTypes = {  
52    filterMovies: PropTypes.func.isRequired,  
53  };  
54  
55  const mapDispatchToProps = (dispatch) => {  
56    return {  
57      filterMovies: (text) => {  
58        dispatch(filterMovies(text));  
59      },  
60    };  
61  };  
62  
63  export default connect(  
64    null,  
65    mapDispatchToProps,  
66  })(FilterMovies);  
67
```

App- presentation. jsx

```
39 const mapStateToProps = state => {
40   const allMovies = state.movies.movies;
41   let filteredMovies;
42
43   if (state.movies.searchText) {
44     const searchLower = state.movies.searchText.toLowerCase();
45     filteredMovies = allMovies.filter(m => m.title.toLowerCase().indexOf(searchLower) !== -1);
46
47     if (!filteredMovies.length) {
48       filteredMovies = null;
49     }
50   }
51
52   return {
53     user: state.user,
54     playing: state.movies.playing,
55     movies: filteredMovies || allMovies,
56   };
57 };
58
59 export default connect(mapStateToProps)(AppPresentation);
```

Cleanup

- Header.jsx

- Main



Remove app-container

The AppContainer is no longer needed.

Index.js

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3
4  import { createStore, applyMiddleware } from "redux";
5  import { Provider } from "react-redux";
6  import thunk from "redux-thunk";
7
8  import AppPresentation from "../app-presentation";
9
10 import reducers from "../reducers";
11 import { login } from "../actions";
12
13 // npm install bootstrap --save
14 import "../node_modules/bootstrap/dist/css/bootstrap.min.css";
15
16 are(thunk));
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 );
```





Connect only to
required movie data

Main-page

```
1  import React, { PropTypes } from 'react';
2  import Header from './header';
3  import Billboard from './billboard';
4  import GenreList from './genre-list';
5
6  const MainPage = ({ user }) => (
7    <div>
8      <Header user={user} />
9      <Billboard />
10     <GenreList />
11   </div>
12 );
13
14 MainPage.propTypes = {
15   user: PropTypes.object.isRequired,
16 };
17
18 export default MainPage;
```

Genre-list

```
31  GenreList.propTypes = {
32    movies: PropTypes.array.isRequired,
33  };
34
35  const mapStateToProps = state => {
36    const allMovies = state.movies.movies;
37    let filteredMovies;
38
39    if (state.movies.searchText) {
40      const searchLower = state.movies.searchText.toLowerCase();
41      filteredMovies = allMovies.filter(m => m.title.toLowerCase().indexOf(searchLower) !== -1);
42
43      if (!filteredMovies.length) {
44        filteredMovies = null;
45      }
46    }
47
48    return {
49      movies: filteredMovies || allMovies
50    };
51  };
52
53  export default connect(mapStateToProps)(GenreList);
```

Billboard

```
17 Billboard.propTypes = {
18   movie: PropTypes.object.isRequired,
19 };
20
21 const mapStateToProps = state => {
22   const allMovies = state.movies.movies;
23   let firstMovies;
24
25   if (state.movies.searchText) {
26     const searchLower = state.movies.searchText.toLowerCase();
27     firstMovies = allMovies.find(m => m.title.toLowerCase().indexOf(searchLower) !== -1);
28   }
29
30   return {
31     movie: firstMovies || allMovies[0]
32   };
33 };
34
35 export default connect(mapStateToProps)(Billboard);
36
```

App- presentation

```
8 | const AppPresentation = ({ user, hasMovies, playing }) => {
9 |   let component = null;
10 |
11 |   if (!user) {
12 |     component = <LoginPage />;
13 |   } else if (!hasMovies) {
14 |     component = <AjaxLoading />;
15 |   } else if (playing) {
16 |     component = <PlayingMovie />;
17 |   } else {
18 |     component = (
19 |       <MainPage
20 |         user={user}
21 |       />
22 |     );
23 |   }
24 |
25 |   return (
26 |     <div className="container">
27 |       {component}
28 |     </div>
29 |   );
30 | };
31 |
32 | AppPresentation.propTypes = {
33 |   user: PropTypes.object,
34 |   hasMovies: PropTypes.bool,
35 |   playing: PropTypes.object,
36 | };
37 |
38 | const mapStateToProps = state => {
39 |   return {
40 |     user: state.user,
41 |     playing: state.movies.playing,
42 |     hasMovies: !!state.movies.movies
43 |   };
44 | };
```

Bonus Exercises

- Extract updating localStorage from user reducer and do this with middleware
- The user is passed from AppPresentation => MainPage => Header

Maurice de Beijer

@mauricedb

maurice.de.beijer
@gmail.com

