The action creators dosen’t return an action but a function. The Middleware (Thunk) invokes the function that contains the axios call to the API. When the response is ready from the API call – the dispatch function is invoked with an action-object containing the data as payload

import axios from 'axios';

export default axios.create({

    baseURL: 'http://jsonplaceholder.typicode.com'

});

import { createStore, **applyMiddleware** } from 'redux';

**import thunk from 'redux-thunk'**;

import App from './components/App';

import reducers from './reducers';

const store = createStore(reducers, **applyMiddleware(thunk));**

ReactDOM.render(

    <Provider store={store}>

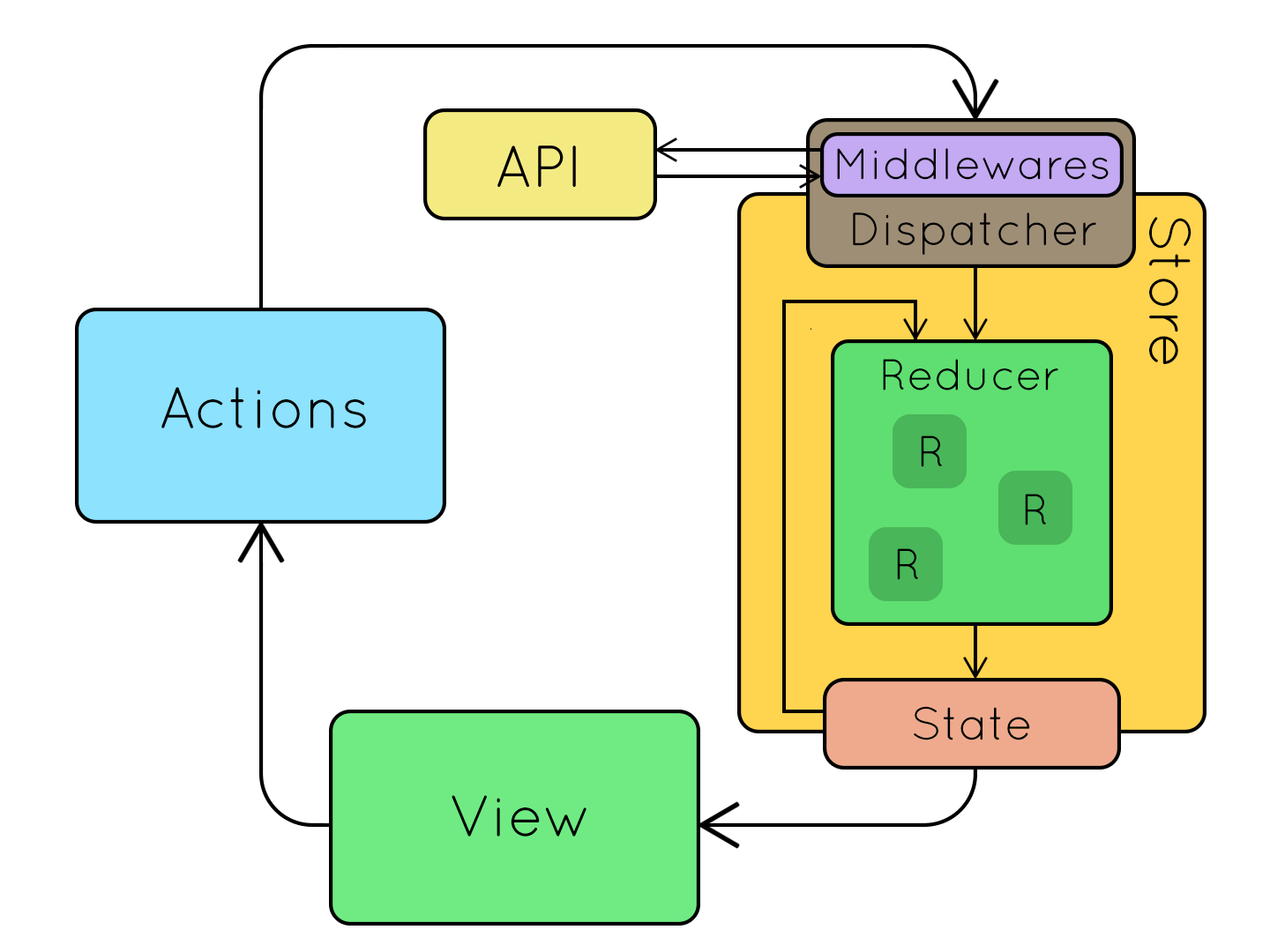
            <App />

    </Provider>,

     document.getElementById('root')

);

React-Redux-Thunk Overview (Blog App)



Notice: the example are using Lodash: <https://lodash.com/>

C/Henrik Høltzer Zealand 16.3 2020 (Image source: <http://slides.com/jenyaterpil/redux-from-twitter-hype-to-production>)

export const fetchPosts = () => async dispatch => {

        const response = await jsonPlaceholder.get('/posts');

        dispatch({type: 'FETCH\_POSTS', payload: response.data})

};

export const fetchUser = id => dispatch => { \_fetchUser(id, dispatch);};

const \_fetchUser = \_.memoize(async (id, dispatch) => {

    const response = await jsonPlaceholder.get(`/users/${id}`);

    dispatch({type: 'FETCH\_USER', payload: response.data})

});

class App extends Component {

  render() {

    return (

      <div className="ui container">

        <header className="App-header">

          <PostList/>

        </header>

      </div>

    );

  }

}

export default combineReducers({

    posts : postsReducer,

    users : usersReducer

});

export default (state=[], action) => {

   switch (action.type) {

       case 'FETCH\_POSTS': return action.payload;

       default: return state;

   }

};

export default (state=[], action) => {

    switch (action.type) {

        case 'FETCH\_USER' : return [...state, action.payload];

        default: return state;

    }

}

class UserHeader extends React.Component {

    componentDidMount(){

        this.props.fetchUser(this.props.userId);

    }

    render() {

        const { user } = this.props;

        if (!user){return null;}

        return <div className="header">{user.name}</div>;

    }

}

const mapStateToProps = (state, ownProps) => { return { user: state.users.find(user => user.id === ownProps.userId) };};

export default connect(mapStateToProps, {fetchUser})(UserHeader);

class PostList extends React.Component {

    componentDidMount(){

        this.props.fetchPosts();

    }

    renderList() {

        return this.props.posts.map(post => {

            return (

                <div className="item" key={post.id}>

                    <i className="large middle aligned icon user" />

                    <div className="content">

                        <div className="description">

                            <h2>{post.title}</h2>

                            <p>{post.body}</p>

                        </div>

                        <UserHeader userId={post.userId}/>

                    </div>

                </div>

            )

        })

    }

    render() {return <div className="ui relaxed diveded list">{this.renderList()}</div>;}

}

const mapStateToProps = state => {return { posts: state.posts };}

export default connect(mapStateToProps, {fetchPosts})(PostList);