Malvika Sriram

04/17/17

COMP 129

Research Spike

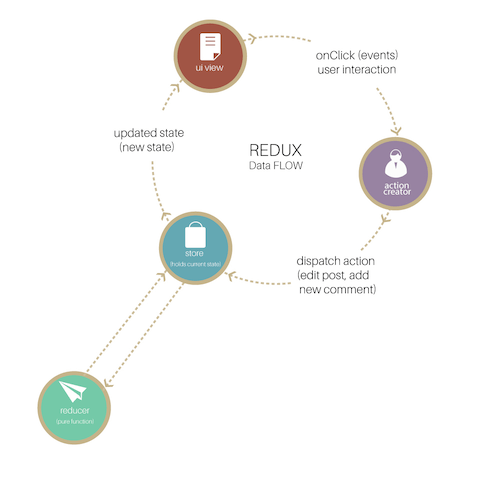
Redux Basics Notes

-Redux manages states in components, and this helps integrate one page applications in a much smarter way.

-Managing multiple states are difficult, but redux replaces all of these states as one “state tree”, or global state.

-State tree is an object, which cannot change directly as it is immutable. Actions and Reducers create a new object when something has been changed.

-Components of Redux: Action, Store, Reducer



Source: <https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/>

Actions: Events sending data from an application to the store.

Store: The immutable global state is held in here, and it also provides easier state success (methods)

Reducers: Update the existing state by using current state and action (pure functions)

-Pure functions return new values instead of changing an existing object.

-Benefits:

-Reusable code and efficiency

-recursion and passing functions as arguments

-Main global state is immutable

-Emphasis on Functional Programming

Example: Creating redux component for our current project with guidance from a team member with existing redux knowledge:

**Profile index.js=component code (usage of redux in bold)**

import React, { PropTypes, Component } from ‘react’;

import { client } from ‘../../modules/api-client’;

class Header extends Component{

render() {

return (

<div className=’profile-header’>

<h3>My Profile</h3>

</div>

);

}

}

export default class Profile extends Component {

constructor(){

super();

}

render() {

return (

<div className=”content”>

<Header></Header>

<div className=’profile-container’>

<div >

<div className=’profile-avatar’>

<img src={ **this.props.profile**.avatar } alt=”User Image”/>

</div>

<div >

<p className=’profile-name’ >{**this.props.profile**.name} </p>

<p className=’profile-handle’ > {**this.props.profile**.handle} </p>

</div>

</div>

<div className=’profile-otherinfo’>

<p> Phone : {**this.props.profile**.phoneNumber} </p>

<p> Birthday : {**this.props.profile**.dob} </p>

<p> Where I Live : {**this.props.profile**.address} </p>

<div>

<p>Friends </p>

<div className=’profile-friendlist’>

{ **this.props.pofile**.friends.map((friend, k) => {

return <img src={ friend } key={k} alt=”User Image”/>

})}

</div>

</div>

</div>

</div>

</div>

);

}

}

**Profile Redux Container(for this.props.[componentname])**

import Profile from '../components/Profile/index';

import { connect } from 'react-redux';

function mapStateToProps(state, ownProps){

return {

profile: state.profile

};

}

export default connect(mapStateToProps, null)(Profile);

**Profile Reducer with initial state from index.js**

import {

PROFILE\_GET\_USER,

PROFILE\_GET\_USER\_SUCCESS,

PROFILE\_GET\_USER\_FAILURE,

PROFILE\_SET\_NAME,

PROFILE\_SET\_AVATAR

} from '../actions/profile';

const INITIAL\_STATE = {

name:'Abc',

handle:'@abc',

avatar:'/img/avatar1.jpg',

phoneNumber:'+1-2345678',

dob:'12-01-2011',

address:'Here and There',

friends:['/img/avatar1.jpg','/img/avatar2.jpg','/img/avatar3.jpg']

};

export default function(state = INITIAL\_STATE, action){

let user;

switch (action.type){

case PROFILE\_GET\_USER:

return {

...state,

isLoading: true,

error: ''

};

break;

case PROFILE\_GET\_USER\_SUCCESS:

return {

...state,

isLoading: false,

error: ''

};

break;

case PROFILE\_GET\_USER\_FAILURE:

return {

...state,

isLoading: false,

error: action.payload

};

break;

case PROFILE\_SET\_NAME:

user.name = action.payload.name;

return {

...state,

user

};

brSourceseak;

case PROFILE\_SET\_AVATAR:

user.avatar = actionpayload.avatar;

return {

...state,

user

};

break;

}

return state;

};

Sources:

1. <https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/>
2. <http://redux.js.org/docs/introduction/Motivation.html>
3. <https://voice.kadira.io/let-s-compose-some-react-containers-3b91b6d9b7c8>