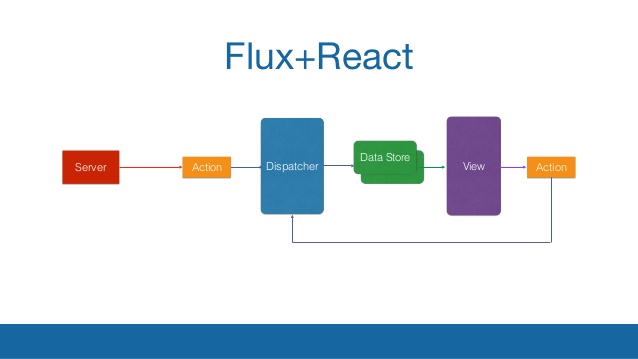


Getting Started with Flux

Flux is an architecture which is similar to the Publisher / Subscriber pattern. To understand it simply:

1. A component publishes events using Action Creators
2. Action Creator dispatches events to store using a Dispatcher
3. Store registers for events which are dispatched
4. Store updates its internal data structure with any changes required and emits a change event
5. Components subscribe to change events emitted by the store to modify its internal state and re-render accordingly based on the data returned from the store.

Note that there can be multiple store's listening to Events dispatched from Action Creators and similarly multiple components can register for updates from the store and re-render accordingly.



**App.js**

import React from "react";

import Content from './Content'

function App() {

return (

<>

<div> <Content text="A simple flux implementation with React" /> </div>

</>

);

}

export default App;

**Content.js**

import React from 'react'

import Button from './Button';

import List from './List'

import AppActions from './AppActions';

import AppStore from './AppStore'

class Content extends React.Component {

constructor(props) {

super(props);

this.state = { articles: [], articlesApproved: [], message: '' };

this.handleClick = this.handleClick.bind(this);

this.onSubmit = this.onSubmit.bind(this);

this.onRemove = this.onRemove.bind(this);

}

handleClick() {

if (document.getElementById('simpletext').value.length > 0 && this.state.articles.length < 10) { AppActions.submitArticle(document.getElementById('simpletext').value)

document.getElementById('simpletext').value = ''

}

}

componentDidMount() {

AppStore.addChangeListener('STORE\_SUBMIT\_ARTICLE', this.onSubmit);

AppStore.addChangeListener('STORE\_REMOVE\_ARTICLE', this.onRemove);

}

onRemove() {

this.listArticles() }

onSubmit() {

this.listArticles() }

listArticles()

{ let usermessage = ''

if (this.state.articles.length > 9) {

usermessage = 'You have exceeded the number of articles you can submit,You cannot add more articles'

}

this.setState({

articles: AppStore.getAll(),

articlesApproved: AppStore.getApproved(),

message: usermessage

})

}

componentWillUnmount() {

AppStore.removeChangeListener('STORE\_SUBMIT\_ARTICLE', this.onChange)

AppStore.removeChangeListener('STORE\_REMOVE\_ARTICLE', this.onRemove)

}

render() {

var simpleContent =

<div>

{this.props.text}

<br />

Enter text : <input type="text" name="simpletext" id="simpletext" />

<Button handleClick={this.handleClick} text="SUBMIT" />

<br />

<List articles={this.state.articles} listHeader="Submitted Articles" />

{this.state.message}

<List articles={this.state.articlesApproved} listHeader="Approval Status" />

</div>;

return simpleContent;

}

}

export default Content;

**List.js**

import React from 'react'

import Button from './Button'

import AppActions from './AppActions'

class List extends React.Component {

handleClick(key) {

AppActions.removeArticle(key)

}

render() {

var articles = this.props.articles !== undefined ? this.props.articles.map((article,i) => { return <li key={i}> Article {i+1}:{article} <Button handleClick={()=>this.handleClick(i)} text="X"/></li>

}) :[];

return (

<div>

<h1>{this.props.listHeader}</h1>

<ul>

{articles}

</ul>

</div>

);

}

}

export default List;

**Button.js**

import React from 'react'

const Button = (props) => <button onClick={props.handleClick} >{props.text}</button>

export default Button;

**appDispatcher.js**

import { Dispatcher } from "flux";

const dispatcher = new Dispatcher();

export default dispatcher;

**AppActions.js**

import AppDispatcher from './appDispatcher';

class AppActions {

submitArticle(data) {

AppDispatcher.dispatch({

actionType: 'SUBMIT\_ARTICLE',

value: data

});

AppDispatcher.dispatch({

actionType: 'APPROVE\_ARTICLE',

value: data

});

}

removeArticle(key)

{

AppDispatcher.dispatch({

actionType: 'REMOVE\_ARTICLE',

value: key

});

}

}

export default new AppActions()

**AppStore.js**

import AppDispatcher from './appDispatcher';

import { EventEmitter } from 'events';

let \_articles = [];

let \_articlesApproved = []

class AppStore extends EventEmitter {

constructor() {

super();

this.dispatchToken = AppDispatcher.register(this.dispatcherCallback.bind(this))

}

emitChange(eventName) {

this.emit(eventName);

}

getAll() {

return \_articles;

}

getApproved() {

return \_articlesApproved;

}

submitArticle(article) {

\_articles.push(article);

}

removeArticle(key)

{

\_articles.splice(key,1);

\_articlesApproved.splice(key,1)

}

approveArticle(article) {

if (article.length <= 10) {

\_articlesApproved.push('[Approved]:' + article);

}

else {

\_articlesApproved.push('[Rejected]:' + article);

}

}

addChangeListener(eventName, callback) {

this.on(eventName, callback);

}

removeChangeListener(eventName, callback) {

this.removeListener(eventName, callback);

}

dispatcherCallback(action) {

// eslint-disable-next-line

switch (action.actionType) {

case 'SUBMIT\_ARTICLE':

this.submitArticle(action.value);

break;

case 'APPROVE\_ARTICLE':

this.approveArticle(action.value);

break;

case 'REMOVE\_ARTICLE':

this.removeArticle(action.value);

}

this.emitChange('STORE\_' + action.actionType);

return true;

}

}

export default new AppStore();