[TOC]

使用react仿简书的项目

1. 安装依赖

`npm install`

2. 启动项目

`npm start`

3. 打包项目

`npm run build`

4. 使用的插件以及版本

```
    "axios": "^0.19.0",
    "immutable": "^3.8.2",
    "react": "^16.8.6",
    "react-dom": "^16.8.6",
    "react-redux": "^5.0.7",
    "react-router-dom": "^4.3.1",
    "react-scripts": "3.0.1",
    "react-transition-group": "^2.3.1",
    "redux": "^4.0.0",
    "redux-immutable": "^4.0.0",
    "redux-thunk": "^2.3.0",
    "styled-components": "^3.3.2"
```

5. 项目说明：

这是仿简书的部分功能的一个React项目，这个React的功能不是很多，但是涵盖的实现和技术还是蛮丰富的，希望大家多多指教，喜欢的小伙伴可以动个小手点个star。整个项目的开发流程我都使用markdown进行了点滴记录。

6. 项目开发所用技术说明

- 既然是一个React的项目，那么`react`和`react-dom`肯定都是必不可少的，但是我使用的版本是最新的`16.8.6`
- `axios`是一个基于promise的HTTP的库，用途广泛，体积小，使用简单，在项目中向后台发送请求使用的就是axios这个库。使用的版本是`0.19.0`
- `styled-components`充当'css'的角色，使用的版本是`3.3.2`
- `redux`在这边我就不多说了，具体可以参照我的博客 ，使用的版本是`4.0.0`
- `react-redux`是一个封装的库，为了提高react和redux开发的便利, 使用的版本是`5.0.7`
- `react-transition-group`是react动画库，使用的版本是`2.3.1`

- `immutable`数据的不可变
- `redux-immutable`一个插件库，redux和immutable的协调开发
- `redux-thunk`一个react的中间件
- `react-router-dom` react的插件

7. 开发流程

1.首先是使用`reset.css`对各个浏览器的样式做出一些调整,`style.js`文件，这个是全局样式文件。reset

```
import { injectGlobal } from 'styled-components';

injectGlobal`
        html, body, div, span, applet, object, iframe,
        h1, h2, h3, h4, h5, h6, p, blockquote, pre,
        a, abbr, acronym, address, big, cite, code,
        del, dfn, em, img, ins, kbd, q, s, samp,
        small, strike, strong, sub, sup, tt, var,
        b, u, i, center,
        dl, dt, dd, ol, ul, li,
        fieldset, form, label, legend,
        table, caption, tbody, tfoot, thead, tr, th, td,
        article, aside, canvas, details, embed,
        figure, figcaption, footer, header, hgroup,
        menu, nav, output, ruby, section, summary,
        time, mark, audio, video {
                margin: 0;
                padding: 0;
                border: 0;
                font-size: 100%;
                font: inherit;
                vertical-align: baseline;
        }
        /* HTML5 display-role reset for older browsers */
        article, aside, details, figcaption, figure,
        footer, header, hgroup, menu, nav, section {
                display: block;
        }
        body {
                line-height: 1;
        }
        ol, ul {
                list-style: none;
        }
        blockquote, q {
                quotes: none;
        }
        blockquote:before, blockquote:after,
        q:before, q:after {
                content: '';
                content: none;
        }
        table {
                border-collapse: collapse;
                border-spacing: 0;
    }
`;
```

2. 入口文件(index.js)

```
import React from 'react';
import ReactDOM from 'react-dom';
import './style.js';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

3. 程序的启动点(App.js)

> - 这边使用`react-redux`提供的组件`Provider`作为根组件，这样会使得里面的所有的组件或者是内部的组件的组件都可以使用`react-redux`带来的便利性，数据的共享
> - `store`总仓库，并且绑定到`provider`组件上
> - 使用`BrowserRouter`组件来做路由控制
> - 使用`Route`来设置组件，`exact`绝对路由路径匹配，`component`设置对应的组件，`path`路由的名称
>
>   注意其中一个路由的`path="/detail/:id"`这个是动态路由

```
import React, { PureComponent, Fragment } from 'react';
import { Provider } from 'react-redux'
import store from './store'
import { BrowserRouter, Route } from 'react-router-dom'
import Header from './common/header'
import Home from './pages/home'
import Detail from './pages/detail'
import Login from './pages/login'

class App extends PureComponent {
  render() {
    return (
      <Provider store={store}>
        <Fragment>
          <BrowserRouter>
            <div>
              <Header />
              <Fragment>
                <Route path="/" exact component={Home}></Route>
                <Route path="/detail/:id" exact component={Detail}></Route>
                <Route path="/login" exact component={Login}></Route>
              </Fragment>
            </div>
          </BrowserRouter>
        </Fragment>
      </Provider>

    )
  }
}
export default App;
```

4. 全局redux(index.js)（全局store）

> 1. 引入`redux`, 使用其中的组件`createStore`, `compose`, `applyMiddleware`
> 2. 引入`reducer`
> 3. 引入中间件`redux-thunk`
> 4. 默认导出`store`

```
import { createStore, compose, applyMiddleware } from 'redux'
import reducer from './reducer'
import thunk from 'redux-thunk'

const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;
const store = createStore(reducer, composeEnhancers(
   applyMiddleware(thunk)
));

export default store;
```

5. 全局reducer(reducer.js)(总图书管理员)

> 这边使用`redux-immutable`来管理各个`reducer`， 并且给各个`reducer`取个别名来方便从`store`中取值。并且导出`reducer`, 首先需要说明的是什么呢？为什么需要使用集中管理这个词，为什么不把所有的`reducer`的都放到这一个`reducer`中呢，其实这样做的目的是为了项目的可维护性，可管理性，以及问题的可排查性。
>
> 将各个`reducer`分配到各个组件，单独管理。

```
import { combineReducers } from 'redux-immutable'
import { reducer as headerReducer } from '../common/header/store'
import { reducer as homeReducer } from '../pages/home/store'
import { reducer as detailReducer } from '../pages/detail/store'
import { reducer as loginReducer } from '../pages/login/store'

const reducer = combineReducers({
   header: headerReducer,
   home: homeReducer,
   detail: detailReducer,
   login: loginReducer
})

export default reducer
```

6. 编写header部分组件

> 1. index.js
> 2. index.js(store)
> 3. reducer.js(store)
> 4. actionCreators.js(store)
> 5. constants(store)

6.1 入口文件(index.js)

```
import React, { PureComponent } from 'react'
import { connect } from 'react-redux'
import { CSSTransition } from 'react-transition-group'
import { actionCreators } from './store'
import { Link } from 'react-router-dom'
import { actionCreator  as loginActionCreator } from '../../pages/login/store'
import {
   HeaderWrapper,
   Logo,
   Nav,
   NavItem,
   NavSearch,
   Addition,
   Button,
   SearchWrapper,
   SearchInfo,
   SearchInfoTitle,
   SearchInfoSwitch,
   SearchInfoItem,
   SearchInfoList
} from './styles'

class Header extends PureComponent {

   getListArea = () => {
      const { focused, list, page, totalPage, mouseIn, handleMouseEnter,
handleMouseLeave, hanleChangePage } = this.props
      const newList = list.toJS()
      const pageList = []
      if (newList.length) {
         for (let i = (page - 1) * 10; i < page * 10; i++) {
            pageList.push(
               <SearchInfoItem key={newList[i]}>{newList[i]}</SearchInfoItem>
```

```
                )
            }
        }

        if (focused || mouseIn) {
            return (
                <SearchInfo
                    onMouseEnter={handleMouseEnter}
                    onMouseLeave={handleMouseLeave}
                >
                    <SearchInfoTitle>
                        热门搜索
                        <SearchInfoSwitch
                            onClick={() => { hanleChangePage(page, totalPage) }}
                        >
                            <i className='spin iconfont iconspin' style={{ fontSize: 12
}}></i>
                            换一批
                        </SearchInfoSwitch>
                    </SearchInfoTitle>
                    <SearchInfoList>
                        {pageList}
                    </SearchInfoList>
                </SearchInfo>
            )
        } else {
            return null
        }
    }
    render() {
        const { focused, list, handleInputFocus, handleInputBlur, handleDownLoad,
login, logout } = this.props
        return (
            <HeaderWrapper>
                <Link to="/">
                    <Logo />
                </Link>
                <Nav>
                    <Link to="/">
                        <NavItem className='left active'>首页</NavItem>
                    </Link>
                    <NavItem className='left download' onClick={() => {
handleDownLoad() }}>下载APP</NavItem>
                    {
                        login ?
                            <NavItem className='right' onClick={logout}>退出</NavItem> :
                            <Link to="/login"><NavItem className='right'>登录</NavItem>
</Link>
                    }
                    <NavItem className='right'>
                        <i className='iconfont iconAa' style={{ fontSize: 25 }}></i>
                    </NavItem>
                    <SearchWrapper>
                        <CSSTransition
                        in={this.props.focused}
                        timeout={500}
                        classNames='slide'
                     >
                        <NavSearch
                           className={focused ? 'focused' : ''}
                           onFocus={() => { handleInputFocus(list) }}
                           onBlur={() => { handleInputBlur() }}
                        ></NavSearch>
                     </CSSTransition>
                     <i className={focused ? 'focus-icon iconfont iconxiazai17 zoom'
: 'iconfont iconxiazai17 zoom'}></i>
                     {this.getListArea()}
                  </SearchWrapper>

                  <Addition>
                     <Button className='writting'>
                        <i className='iconfont iconpen' style={{ fontSize: 12,
marginRight: 5, }}></i>
                        写文章</Button>
                     <Button className='reg'>注册</Button>
                  </Addition>
               </Nav>
            </HeaderWrapper>
         )
   }
}

const mapStateToProps = (state) => {
   return {
      focused: state.get('header').get('focused'),
      list: state.get('header').get('list'),
      mouseIn: state.get('header').get('mouseIn'),
      page: state.get('header').get('page'),
      totalPage: state.get('header').get('totalPage'),
      login: state.get('login').get('login')
   }
}

const mapDispatchToProps = (dispatch) => {
   return {
      handleInputFocus(list) {
         if (list.size === 0) {
            dispatch(actionCreators.getList())
         }
         dispatch(actionCreators.searchFocus())
      },
      handleInputBlur() {
         dispatch(actionCreators.searchBlur())
      },
      handleMouseEnter() {
         dispatch(actionCreators.mouseEnter())
      },
      handleMouseLeave() {
            dispatch(actionCreators.mouseLeave())
        },
        hanleChangePage(page, totalPage) {
            console.log(page, totalPage)
            if (page < totalPage) {
                dispatch(actionCreators.changePage(page + 1))
            } else {
                dispatch(actionCreators.changePage(1))
            }
        },
        handleDownLoad() {
            window.location.href = "https://www.jianshu.com/apps?
utm_medium=desktop&utm_source=navbar-apps"
        },
        logout() {
            dispatch(loginActionCreator.logout())
        }
    }
}
export default connect(mapStateToProps, mapDispatchToProps)(Header)
```

6.2 index.js(store)

```
import reducer from './reducer'
import * as actionCreators from './actionCreators'
import * as constants from './constants'

export {
    reducer,
    actionCreators,
    constants
}
```

6.3 reducer.js(store)

```
import * as constants from './constants'
import { fromJS } from 'immutable'

const defaultState = fromJS({
    focused: false,
    list: [],
    mouseIn: false,
    page: 1,
    totalPage: 1
})

export default (state = defaultState, action) => {
    switch (action.type) {
        case constants.SEARCH_FOCUS:
            return state.set('focused', true)
        case constants.SEARCH_BLUR:
            return state.set('focused', false)
        case constants.CHANGE_LIST:
            return state.merge({
                list: action.data,
                totalPage: action.totalPage
            })
        case constants.MOUSE_ENTER:
            return state.set('mouseIn', true)
        case constants.MOUSE_LEAVE:
            return state.set('mouseIn', false)
        case constants.CHANGE_PAGE:
            return state.set('page', action.page)
        default:
            return state
    }
}
```

6.4 actionCreators.js(store)

```
import * as constants from './constants'
import axios from 'axios'
import { fromJS } from 'immutable'

export const searchFocus = () => ({
    type: constants.SEARCH_FOCUS
})

export const searchBlur = () => ({
    type: constants.SEARCH_BLUR
})

export const mouseEnter = () => ({
    type: constants.MOUSE_ENTER
})

export const mouseLeave = () => ({
    type: constants.MOUSE_LEAVE
})

export const changePage = (page) => ({
    type: constants.CHANGE_PAGE,
    page: page
})


export const getList = () => {
    return (dispatch) => {
      axios.get('/api/headerList.json').then((res) => {
        const data = res.data
        dispatch(changeList(data.data))
      }).catch((error) => {
        console.log(error)
      })
    }
  }

  const changeList = (data) => ({
    type: constants.CHANGE_LIST,
    data: fromJS(data),
    totalPage: Math.ceil(data.length / 10)

  })
```

6.5 contants.js(store)

```
  export const SEARCH_FOCUS = 'header/SEARCH_FOCUS'
  export const SEARCH_BLUR = 'header/SEARCH_BLUR'
  export const CHANGE_LIST = 'header/CHANGE_LIST'
  export const MOUSE_ENTER = 'header/MOUSE_ENTER'
  export const MOUSE_LEAVE = 'header/MOUSE_LEAVE'
  export const CHANGE_PAGE = 'header/CHANGE_PAGE'
```