**"devDependencies"**: {  
 **"react-scripts"**: **"^0.6.0"**,  
 **"redux-logger"**: **"^2.6.1"**},  
**"dependencies"**: {  
 **"react"**: **"^15.3.0"**,  
 **"react-dom"**: **"^15.3.0"**,  
 **"react-redux"**: **"^4.4.5"**,  
 **"redux"**: **"^3.5.2"**,  
 **"redux-thunk"**: **"^2.1.0"**},  
**"scripts"**: {  
 **"start"**: **"react-scripts start"**,  
 **"build"**: **"react-scripts build"**,  
 **"eject"**: **"react-scripts eject"**}

//////////////////////////////////////////////////////

.gitignore

//////////////////////////////////////////////////////

# dependencies  
node\_modules  
  
# production  
build

# misc  
.DS\_Store  
npm-debug.log

/////////////////////////////////////////////////////////////////////

index.js store

/////////////////////////////////////////////////////////////////////

**import** React **from 'react'**;  
**import** { render } **from 'react-dom'**;  
**import** { createStore, *applyMiddleware* } **from 'redux'**;  
**import** { **Provider** } **from 'react-redux'**;  
**import** thunk **from 'redux-thunk'**;  
**import** createLogger **from 'redux-logger'**;  
**import** reducer **from './reducers'  
import** App **from './containers/App'  
  
const** middleware = [thunk];  
  
**if**(process.env.**NODE\_ENV** !== **'production'**){  
 middleware.push(createLogger())  
}  
  
**const** store = createStore(  
 reducer,  
 *applyMiddleware*(...middleware)  
)  
  
render(  
 <**Provider store=**{store}>  
 <**App**/>  
 </**Provider**>,  
 **document**.getElementById(**'root'**)  
)

/////////////////////////////////////////////////////////////////////

最基础的两个组件

/////////////////////////////////////////////////////////////////////

**import** React, { PropTypes } **from 'react'**;  
  
**const** *Picker* = ({value, onChange, options})=> (  
 <**span**>  
 <**h1**>{value}</**h1**>  
 <**select onChange=**{e=>onChange(e.**target**.**value**)} **value=**{value}>  
 {  
 options.**map**(option=>  
 <**option value=**{option} **key=**{option}>  
 {option}  
 </**option**>)  
 }  
 </**select**>  
 </**span**>  
)  
  
*Picker*.**propTypes** = {  
 **options**: PropTypes.*arrayOf*(  
 PropTypes.**string**.**isRequired** ).**isRequired**,  
 **value**: PropTypes.**string**.**isRequired**,  
 **onChange**: PropTypes.**func**.isRequired  
}  
  
**export default** *Picker*

*////////////////////////////////////////////////////////////***import** React, { PropTypes } **from 'react'**;  
  
**const** *Posts* = ({*posts*}) => {  
 **return** (  
 <**ul**>  
 {*posts*.**map**((post,i)=>  
 <**li key=**{i}>{post.**title**}</**li**>  
 )}  
 </**ul**>  
 )  
}  
 *Posts*.**propTypes** = {  
 **posts**: PropTypes.**array**.**isRequired**}  
  
**export default** *Posts*

/////////////////////////////////////////////////////////////////////

主元件 App.js

/////////////////////////////////////////////////////////////////////

首先要解决需要的这四个属性

{value, onChange, options} {*posts*}

1. 放在开头的PropTypes定义

**static** *propTypes* = {  
 **selectedReddit**: PropTypes.**string**.**isRequired**,  
 **posts**: PropTypes.**array**.**isRequired**,  
 **isFetching**: PropTypes.**bool**.**isRequired**,  
 lastUpdated: PropTypes.**number**,  
 **dispatch**: PropTypes.**func**.isRequired  
}

2. render() 之前dispatch一个action 请求数据

componentDidMount(){  
 **const** { **dispatch**, **selectedReddit** } = **this**.**props  
 dispatch**(*fetchPostsIfNeeded*(**selectedReddit**))  
}

/////////////////////////////////////////////////////////////////////

action/index.js *fetchPostsIfNeeded shouldFetchPosts*

/////////////////////////////////////////////////////////////////////

**export const** *fetchPostsIfNeeded* = reddit => (dispatch, getState) => {  
 **if**(*shouldFetchPosts*(getState(),reddit)){  
 **return** dispatch(*fetchPosts*(reddit))  
 }  
}

**const** *shouldFetchPosts* = (state,reddit) => {  
 **const** posts = state.*postsByReddit*[reddit]

//对state的操作，这里用到了reducer部分  
 **if**(!posts){  
 **return true**;  
 }

//posts为空 需要更新  
 **if**(posts.**isFetching**) {  
 **return false** }

//isFetching is true正在更新  
 **return** posts.**didInvalidate**}

**const** *fetchPosts* = reddit => dispatch => {  
 dispatch(*requestPosts*(reddit))  
 **return** fetch(**`https://www.reddit.com/r/**${reddit}**.json`**)  
 .then(response => response.json())  
 .then(json => dispatch(*receivePosts*(reddit,json)))  
}

**export const** *requestPosts* = reddit => ({  
 **type**: REQUEST\_POSTS,  
 reddit  
})  
  
**export const** *receivePosts* = (reddit,json)=> ({  
 **type**: RECEIVE\_POSTS,  
 reddit,  
 **posts**: json.**data**.**children**.**map**(child => child.**data**),  
 **receiveAt**: Date.now()  
})

/////////////////////////////////////////////////////////////////////

reducer/index.js

/////////////////////////////////////////////////////////////////////

**const** *postsByReddit* = (state={ },action)=>{  
 **switch**(action.**type**) {  
 **case** INVALIDATE\_REDDIT:  
 **case** RECEIVE\_POSTS:  
 **case** REQUEST\_POSTS:  
 **return** {  
 ...state,  
 [action.reddit]: *posts*(state[action.reddit],action)  
 }  
 **default**:  
 **return** state  
 }  
}

具体的posts 定义

**const** *posts* = (state = {  
 **isFetching**: **false**,  
 **didInvalidate**: **false**,  
 **items**: []  
},action)=>{  
 **switch**(action.**type**) {  
 **case** INVALIDATE\_REDDIT:  
 **return** {  
 ...state,  
 **didInvalidate**:**true** }  
 **case** REQUEST\_POSTS:  
 **return** {  
 ...state,  
 **isFetching**:**true**,  
 **didInvalidate**: **false** }  
 **case** RECEIVE\_POSTS:  
 **return** {  
 ...state,  
 **isFetching**:**false**,  
 **didInvalidate**:**false**,  
 **items**:action.**posts**,  
 **lastUpdated**: action.**receiveAt** }  
 **default**:  
 **return** state  
 }  
}

/\*

...state,  
 **isFetching**:**false**,  
 **didInvalidate**:**false**,  
 **items**:action.**posts**,  
 **lastUpdated**: action.**receiveAt**

\*/

//////////////////////////////////////////////////////////////////////////////////

回到App.js组件来

render() {

**const** { **selectedReddit**, **posts**, **isFetching**, **lastUpdated** } = **this**.**props**

**const** isEmpty = **posts**.**length** === 0

//利用这些props state来控制输出

**return** (  
 <**div**>  
 <**Picker value=**{**selectedReddit**}  
 **onChange=**{**this**.*handleChange*}  
 **options=**{[**'reactjs'**,**'frontend'**]}  
 />  
 <**p**>  
 {  
 **lastUpdated** &&  
 <**span**>  
 Last updated at {**new** Date(**lastUpdated**).toLocaleTimeString()}.{**' '**}  
 </**span**>  
 }  
 {  
 !**isFetching** &&  
 <**a href="#"  
 onClick=**{**this**.*handleRefreshClick*}  
 >  
 Refresh  
 </**a**>  
 }  
 </**p**>  
 {isEmpty ?  
 (**isFetching** ? <**h2**>Loading</**h2**> : <**h2**>Empty.</**h2**>)  
 : <**div style=**{{**opacity**:**isFetching** ? 0.5 : 1}}>  
 <**Posts posts=**{**posts**}/>  
 </**div**>  
 }  
 </**div**>  
)

/////////////////////////////////////////////////////////////////////////////////////

重要的一步

**const** *mapStateToProps* = state => {  
 **const** { selectedReddit,postsByReddit } = state  
 **const** {  
 **isFetching**,  
 lastUpdated,  
 items: **posts** } = postsByReddit[selectedReddit] || {  
 **isFetching**: **true**,  
 **items**: []  
 }  
  
 **return** {  
 selectedReddit,  
 **posts**,  
 **isFetching**,  
 lastUpdated  
 }  
}  
**export default connect**(*mapStateToProps*)(App)

接下来是在初次加载完成后

Picker 元件

**const** *Picker* = ({value, onChange, options})=> (  
 。。。

<**select onChange=**{e=>onChange(e.**target**.**value**)} **value=**{value}>

App元件内引用

**onChange=**{**this**.*handleChange*}

*handleChange 函数*

*handleChange*= nextReddit => {  
 **this**.**props**.**dispatch**(*selectReddit*(nextReddit))  
}

////action

**export const** *selectReddit* = reddit => ({  
 **type**: SELECT\_REDDIT,  
 reddit  
})

////reducer

**const** *selectedReddit* = (state = **'reactjs'**, action) => {  
 **switch**(action.**type**) {  
 **case** SELECT\_REDDIT:  
 **return** action.reddit  
 **default**:  
 **return** state  
 }  
}

////////mapStateToProps

**const** { selectedReddit,postsByReddit } = state

**const** {  
 **isFetching**,  
 lastUpdated,  
 items: **posts**} = postsByReddit[selectedReddit] || {  
 **isFetching**: **true**,  
 **items**: []  
}

**return** {  
 selectedReddit,  
 。。。

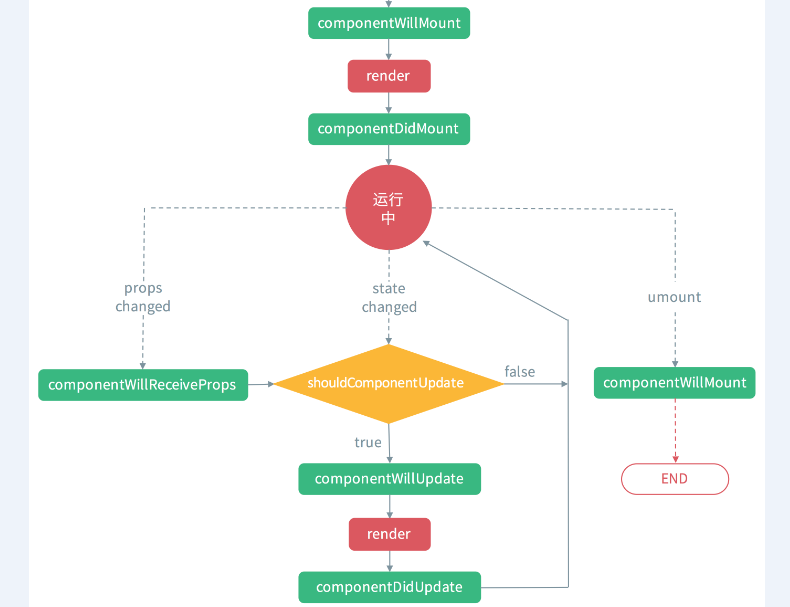
}

**export default connect**(*mapStateToProps*)(App)

此时*selectedReddit改变 被返回 到props*

*元件props的改变将会触发*

*componentWillReceiveProps*



componentWillReceiveProps(nextProps) {  
 **if**(nextProps.selectedReddit !== **this**.**props**.**selectedReddit**){  
 **const** { dispatch, selectedReddit } = nextProps  
 dispatch(*fetchPostsIfNeeded*(selectedReddit))  
 }  
}

接下来的和初始第一次获取一致

//////刷新

初始状态 state

isFetching: **true**,  
**items**: []

在不更新 isFetching = false的时候才显示

{  
 !**isFetching** &&  
 <**a href="#"  
 onClick=**{**this**.*handleRefreshClick*}  
 >  
 Refresh  
 </**a**>  
}

*handleRefreshClick* = e => {  
 e.preventDefault()  
 **const** { **dispatch**, **selectedReddit** } = **this**.**props  
 dispatch**(*invalidateReddit*(**selectedReddit**))  
 **dispatch**(*fetchPostsIfNeeded*(**selectedReddit**))  
}

////action

**export const** *invalidateReddit* = reddit => ({  
 **type**: INVALIDATE\_REDDIT,  
 reddit  
})

**const** *postsByReddit* = (state={ },action)=>{  
 **switch**(action.**type**) {  
 **case** INVALIDATE\_REDDIT:  
 **。。。**

**return** {  
 ...state,  
 [action.reddit]: *posts*(state[action.reddit],action)  
 }  
 **default**:  
 **return** state  
 }  
}

**const** *posts* = (state = {  
 **isFetching**: **false**,  
 **didInvalidate**: **false**,  
 **items**: []  
},action)=>{  
 **switch**(action.**type**) {  
 **case** INVALIDATE\_REDDIT:  
 **return** {  
 ...state,  
 **didInvalidate**:**true** }

// **didInvalidate**:**true 在**

fetchPostsIfNeeded🡪*shouldFetchPosts*

时被用到

**const** *shouldFetchPosts* = (state,reddit) => {  
 **const** posts = state.*postsByReddit*[reddit]  
 **if**(!posts){  
 **return true**;  
 }  
 **if**(posts.**isFetching**) {  
 **return false** }  
 **return** posts.**didInvalidate**}

在更新的情况下 posts不为空，第一个if判断失效

第二个if isFetching = false 也被跳过

因为在每一次 RECEIVE\_POSTS 中，最后返回的

**case** RECEIVE\_POSTS:  
 **return** {  
 ...state,  
 **isFetching**:**false**,  
 **didInvalidate**:**false**,  
 **items**:action.**posts**,  
 **lastUpdated**: action.**receiveAt** }

在 **didInvalidate**:**true** 的情况下，应该更新。

下面关注请求数据过程， 此时isFetching变量一直为true

**export const** *fetchPostsIfNeeded* = reddit => (dispatch, getState) => {  
 **if**(*shouldFetchPosts*(getState(),reddit)){  
 **return** dispatch(*fetchPosts*(reddit))  
 }  
}

*shouldFetchPosts*(getState(),reddit)的值为true 应该进行更新请求数据

dispatch(*fetchPosts*(reddit))

**const** *fetchPosts* = reddit => dispatch => {  
 dispatch(*requestPosts*(reddit))  
 **return** fetch(**`https://www.reddit.com/r/**${reddit}**.json`**)  
 .then(response => response.json())  
 .then(json => dispatch(*receivePosts*(reddit,json)))  
}

dispatch(*requestPosts*(reddit))

//action

**export const** *requestPosts* = reddit => ({  
 **type**: REQUEST\_POSTS,  
 reddit  
})

//reducer

**const** *postsByReddit* = (state={ },action)=>{  
 **switch**(action.**type**) {  
 **case** REQUEST\_POSTS:  
 **return** {  
 ...state,  
 [action.reddit]: *posts*(state[action.reddit],action)  
 }  
 **default**:  
 **return** state  
 }  
}

posts变量

**case** REQUEST\_POSTS:  
 **return** {  
 ...state,  
 **isFetching**:**true**,  
 **didInvalidate**: **false** }

最重要的是isFetching: true

然后回到

**const** *fetchPosts* = reddit => dispatch => {  
 dispatch(*requestPosts*(reddit))  
 **return** fetch(**`https://www.reddit.com/r/**${reddit}**.json`**)  
 .then(response => response.json())  
 .then(json => dispatch(*receivePosts*(reddit,json)))  
}

fetch到数据后转化为json对象，然后把json对象做为参数dispatch一个

dispatch(*receivePosts*(reddit,json)

//action

**export const** *receivePosts* = (reddit,json)=> ({  
 **type**: RECEIVE\_POSTS,  
 reddit,  
 **posts**: json.**data**.**children**.**map**(child => child.**data**),  
 **receiveAt**: Date.now()  
})

//reducer

**const** *postsByReddit* = (state={ },action)=>{  
 **switch**(action.**type**) {  
 **case** RECEIVE\_POSTS:  
 **return** {  
 ...state,  
 [action.reddit]: *posts*(state[action.reddit],action)  
 }  
 **default**:  
 **return** state  
 }  
}

//posts

**case** RECEIVE\_POSTS:  
 **return** {  
 ...state,  
 **isFetching**:**false**,  
 **didInvalidate**:**false**,  
 **items**:action.**posts**,  
 **lastUpdated**: action.**receiveAt** }

//mapStateToProps

**const** {  
 **isFetching**,  
 lastUpdated,  
 items: **posts**} = postsByReddit[selectedReddit] || {  
 **isFetching**: **true**,  
 **items**: []  
}

**return** {  
 selectedReddit,  
 **posts**,  
 **isFetching**,  
 lastUpdated  
}

然后再重新render显示