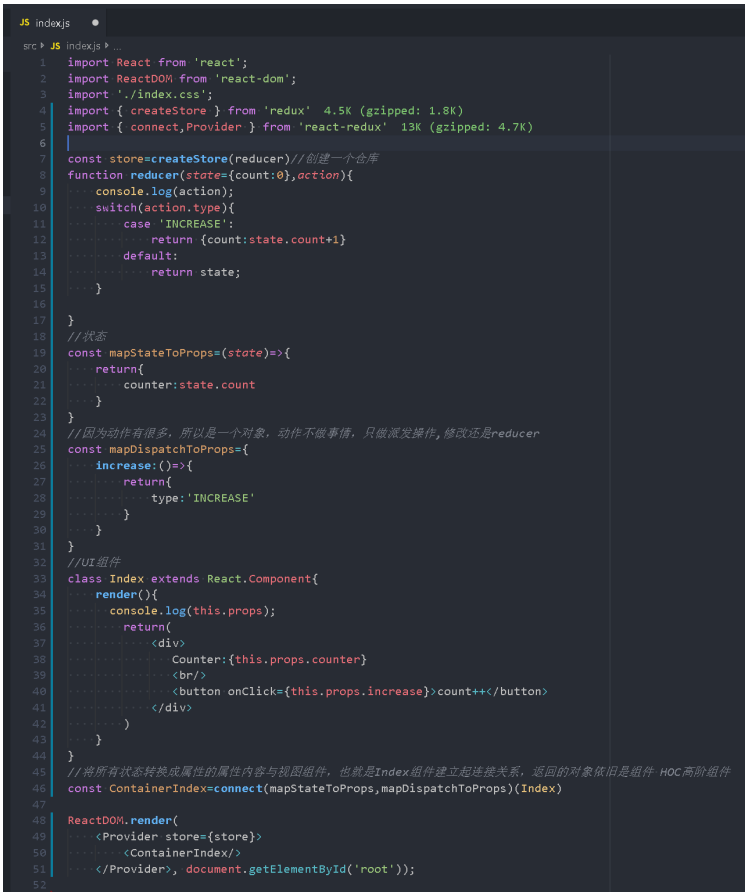
说明：redux只是一个状态管理器，有react-redux,vue-redux等等

1.仓库，万事开头从仓库，因为所有的数据操作都要放在仓库中进行统一管理
2.reducer,因为有了仓库还没有数据，所有的数据都要放在reducer中，reducer是数据中心
3.将reducer放置到store数据仓库中
4.仓库中已经有了state状态管理，但是和UI组件没有关系，因为state属于组件内修改，不可以进行传递
5.需要将状态转换成属性，也就是mapStateToProps
6.状态已经转换成属性，但是和组件没有关系，所以需要将它们两者建立关系
7.通过connect将mapStateToProps和视图组件进行连接，返回来一个容器组件
8.页面渲染的时候就渲染容器组件
9.但是这个时候容器组件并没有和store仓库建立关系
10.通过Provider将store提供给所有组件
11.这样组件就拥有store包含的所有内容，但是我们的仓库里现在仅留一样内容，就是props
12.现在就可以通过this.props获取属性并进行显示操作
13.需要动作进行数据的修改操作，而动作是一系列函数，也和组件没有任何的关系，所以需要将动作转换成属性，mapDispatchToProps
14.注意：mapDispatchToProps只能返回type派发操作,数据的修改还是需要回到reducer数据中心去处理
15.需要通过connect将mapDispatchToProps和视图组件建立连接，这就意味着组件中包含了动作的方法，但是这个动作的方法仍旧是属性
16.视图组件已经不再是单纯的显示功能，它从视图组件已经变成了容器组件，但，仅仅包含属性操作

1.安装

```
yarn add redux react-redux -S
```

2.


index.js
src
index.js
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import { createStore } from 'redux' 4.5K (gzipped: 1.8K)
import { connect,Provider } from 'react-redux' 13K (gzipped: 4.7K)
const store=createStore(reducer)//创建一个仓库
function reducer(state={count:0},action){
console.log(action);
switch(action.type){
case 'INCREASE':
return {count:state.count+1}
default:
return state;
}
}
//状态
const mapStateToProps=(state)=>{
return{
counter:state.count
}
}
//因为动作有很多，所以是一个对象，动作不做事情，只做派发操作,修改还是reducer
const mapDispatchToProps={
increase:()=>{
return{
type:'INCREASE'
}
}
}
//UI组件
class Index extends React.Component{
render(){
console.log(this.props);
return(

Counter:{this.props.counter}

<button onClick={this.props.increase}>count++</button>

)
}
}
//将所有状态转换成属性的属性内容与视图组件，也就是Index组件建立起连接关系，返回的对象依旧是组件 HOC高阶组件
const ContainerIndex=connect(mapStateToProps,mapDispatchToProps)(Index)
ReactDOM.render(
<Provider store={store}>
<ContainerIndex/>
</Provider>, document.getElementById('root'));