import React from ‘react’

import ReactDom from ‘react-dom’

//创建虚拟Dom

const myDiv=React.creatElement(‘div’,{title=’you are a div ’,id:’myDiv’},’

//使用render函数渲染

ReactDom.render(myDiv,document.getElementId(‘App’))

webpack只能默认打包.js后缀名的文件，无法主动处理像.vue .png

const arr=[‘sfjs’,’34434’,’434’,’54564’]

ReactDom.render(<div>

arr.map(item=> <h3 key={item}>{item}</h3>)

</div>,document.getElementId(‘App’))

[(+-/五个符号之一前面的js句子必须要有逗号结尾

jsx语法中 className 代替 class <label>中用htmlFor 代替for

必须要像vue一样 要有唯一根元素包裹

创建组件：

构造函数：组件首字母必须是大写

const dog={

age:10,

height:50

}

function Hello(props){

//props上的属性都是只读的 read-only ,vue也一样

reuturn <div>你好 哈哈哈哈{props.age}</div>

}

ReactDom.render(<div>

//<Hello age={dog.age}></hello>

<Hello {...dog}></hello>

</div>,document.getElementId(‘App’))

如何在导入jsx文件不用输入.jsx后缀名：

在webpack.config.js中加入如下代码：

resolve:{

extensions:[‘.js’,’.jsx’,’.json ’,’.vue’],

alias:{

‘@’:path.join(\_dirname,’./src’)

}

}

第三方 模块配置规则：

module:{

rules:[{

test:/\.js|jsx$/,use:’babel-loader’,exclude:/node\_modules/

},{

test:/\.css$/,use:[‘style-loader’,’css-loader?modules&localIdentName=[path][name]-[local]-[hash:6]’]//moudules:css样式表启用模块化 :global() 可以让模块化失效

}]

}

第2种创建组件的方法:

class Hello extends React.Component{

construtor(){

super();

this.state={//私有数据 可以通过this.state.XXX访问 且 可被修改

}

}

render(){

return <div>第2种创建组件的方法:{this.props.name}</div>

}

}

两种创建组件方式的区别：

都有props，但class创建有自己的生命周期和私有数据

如果要修改state数据 可用 this.setState({},()=>{})方法（异步：提高react底层性能）来修改

import PropTypes from ‘prop-types’

class aa{}

aa.propTypes={

name:PropTypes.func.isrequired,

age:PropTypes.oneOfType([PropTypes.number,PropTypes.string])

}

aa.defaultProps={

name:’你的名字’

}

当一个组件的state props改变 的时候 render函数会重新执行

react生命周期函数：

componentWillMount()

componentDidMount() 仅在挂载完成后执行一次，适合发异步请求

shouldComponentUpdate() 必须返回一个布尔值(false:不走后面所有哦的更新函数)

componentWillUpdate() 在shouldComponentUpdate()后面执行

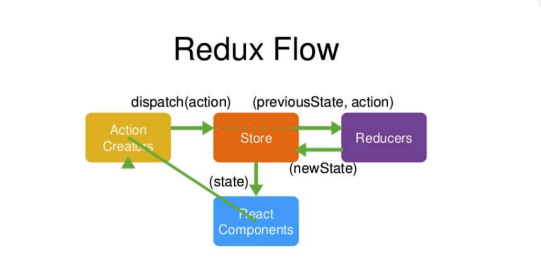
render()此函数必须在存在组件中，其他生命钩子有内置默认函数

componentDidUpdate()

componentWillReceiveProps() 从父组件接收参数 第一次render函数执行存在父组件不执行 已经存在即可执行

componentWillUnmount()

Redux :



index.js:

import {createStore} from ‘redux’

const store=createStore(reducer) //交给reducer处理数据改动逻辑

export default store

reducer.js:

const defaultState={

inputValue:’3232’,

list:[1,2]

}

export default (state=defaultState,action)=>{

return state;

}

组件:

class aa extend Component{

constructor(){

super()}

store.subscribe(()=>{

store.setState(store.getState())

}) //页面监听store数据改好后自动更新

}

action={

type:’xxxx’,

value:’xxxxx’

}

store.dispatch(action) //通知store 页面改动了数据

无状态组件：当一个组件只有render函数，可以转化为一个无状态组件，即为一个函数 返回jsx（节省性能，无周期函数来执行，提升效率）

react 路由：

npm install react-router-dom --save

import {BrowserRouter,Route,Link} from ‘react-router-dom’

<BrowserRouter>

<div>//BrowserRouter里面只能出现一个子节点

<Link to=’/’>首页</Link>

<Route exact path=’/details’ component={DetailsList}>详情</Route>

</div>

</BrowserRouter>