1. 父子组件传值
2. 父组件<Child content={value}/>
3. 子组件可以用this.props.content获取父组件传过来的值。方法也是同样的方法传递。但是父组件需要将传递的方法的作用域绑定到父组件。
4. 子组件如果想要和父组件沟通，需要子组件调用父组件传递过来的方法。

3、



1. 可与其他框架并存，指的是react组件只挂载到一个id上，其他的dom元素可以采用其他的组件（vue, jquery）
2. 单向数据流：指的是数据指向一个方向：父组件可以随时修改子组件的数据，子组件要修改父组件的数据必须调用父组件的方法进行修改。
3. 函数式编程：每个组件由多个函数组成。
4. 渲染render（）
5. 当组件初次创建的时候render函数会执行一次
6. 当state数据发生变化的时候，render函数会被重新执行
7. 当props数据发生变更的时候，render函数会被重新执行
8. ref的使用
9. 在jsx中对应的html标签上增加ref属性其为一个函数，函数的参数就是ref挂载的标签对应的真实的dom对象

<button ref={(button) => {}}>

ref对应的函数会在组件初次渲染的时候执行。所以可以在这个函数中将button对象赋值给组件变量。

1. 如果将ref属性写在自定义的组件上，那个函数的参数就是组件对象的实例。
2. this.setState()

这个函数式异步执行的。

this.setState(obj, fn)

this.setState(fn,fn)

1. 生命周期函数

如果在某个组件中绑定了window事件，那么在离开（移除）此组件的时候也需要把window事件移除，否则会影响其他的组件。

1. react的ajax请求的地址：demo.json

header.json

list.json

ajax请求需要放在componentDidMount中

1. Ant Design组件库：<https://ant.design/index-cn>

npm install antd --save

1. 前端路由

npm install react-router-dom --save

1）index.js中

import {BrowserRouter} from "react-router-dom"

...

ReactDOM.render(

<BrowserRouter>

<App />

</BrowserRouter>,

document.getElementById('root')

);

2）App或者其他任何组件中

import {Routes, Route, Link} from "react-router-dom"

<Routes>

<Route path="/button" element={<ButtonRouter/>}/>

<Route path="/list" element = {<ListRouter/>}/>

</Routes>

1. link用法

import {Link} from "react-router-dom"

render() {

return (

<Link to="/list">

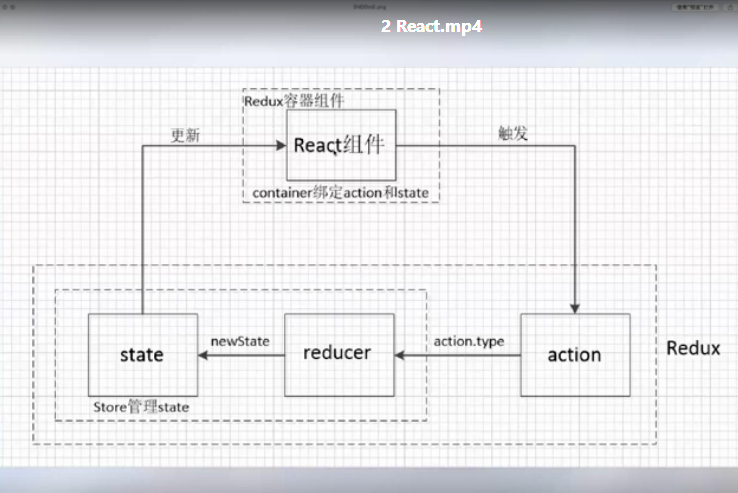
<Button size="large" type="primary">按钮</Button>

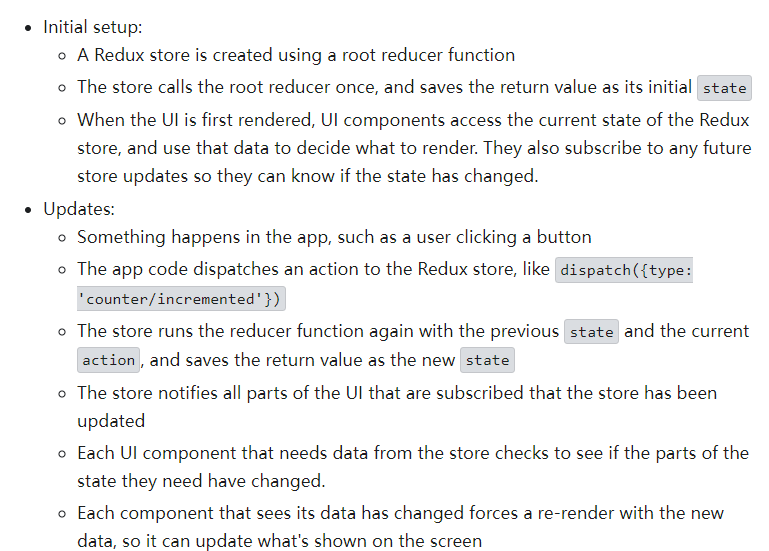
</Link>

)

}

1. redux和react-redux学习





应用中所有的 state 都以一个对象树的形式储存在一个单一的 store 中。 惟一改变 state 的办法是触发 action，一个描述发生什么的对象。 为了描述 action 如何改变 state 树，你需要编写 reducers。

1. 安装： npm install redux react-redux -S

react-redux是连接react视图和redux的中转

2）建立三个文件夹,并在每个文件夹中创建对应的js文件

a、contains

b、actions

**Action** 是把数据从应用（译者注：这里之所以不叫 view 是因为这些数据有可能是服务器响应，用户输入或其它非 view 的数据 ）传到 store 的有效载荷。它是 store 数据的**唯一**来源。

Action 本质上是 JavaScript 普通对象。我们约定，action 内必须使用一个字符串类型的 type 字段来表示将要执行的动作。除了 type 字段外，action 对象的结构完全由你自己决定。**我们应该尽量减少在 action 中传递的数据**。

**Action 创建函数** 就是生成 action 的方法。“action” 和 “action 创建函数” 这两个概念很容易混在一起，使用时最好注意区分。

在 Redux 中的 action 创建函数只是简单的返回一个 action:

function addTodo(text) {

return {

type: ADD\_TODO,

text

}

}

c、reducers/indexjs

**Reducers** 指定了应用状态的变化如何响应 [actions](https://www.redux.org.cn/docs/basics/Actions.html) 并发送到 store 的，记住 actions 只是描述了有事情发生了这一事实，并没有描述应用如何更新 state。

/\*\*

\* 这是一个 reducer，形式为 (state, action) => state 的纯函数。

\* 描述了 action 如何把 state 转变成下一个 state。

\*

\* state 的形式取决于你，可以是基本类型、数组、对象、

\* 甚至是 Immutable.js 生成的数据结构。惟一的要点是

\* 当 state 变化时需要返回全新的对象，而不是修改传入的参数。

\*

\* 下面例子使用 `switch` 语句和字符串来做判断，但你可以写帮助类(helper)

\* 根据不同的约定（如方法映射）来判断，只要适用你的项目即可。

\*/

function counter(state = 0, action) {

switch (action.type) {

case 'INCREMENT':

return state + 1;

case 'DECREMENT':

return state - 1;

default:

return state;

}

}

聚合reducer

import {combineReducers} from "redux"

import user from "./user.js"

export default combineReducers({

user

})

d、store

创建Redux store来存放应用的状态

import userReducer from "../reducer";

import {createStore} from "redux";

export default function userStore(initState) {

return createStore(userReducer, initState)

}

3）异步redux

npm install redux-thunk

a.在store中

// applyMiddleware用来注册中间件

//compose用来聚合中间件和扩展程序，

import {createStore, applyMiddleware,compose} from “redux”

// thunkMiddleware中间件用来拦截actions

import thunkMiddleware from “redux-thunk”

import {createStore} from "redux";

export default function userStore(initState) {

return createStore(userReducer, initState,)

}

b.在actions中

const loginRequest = () => {

return new Promise((resolve) => {

window.setTimeout(() => { //模拟登录完以后执行的动作（ajax请求）

resolve(update({ //update是一个同步的action

isLogin: true,

nickName: “”

}))

}, 3000)

})

}

export const login = (payload) => {

return async (dispatch) => {

const res = await loginRequest();

dispatch(res)

}

}

1. dav学习

1）数据的处理过程

