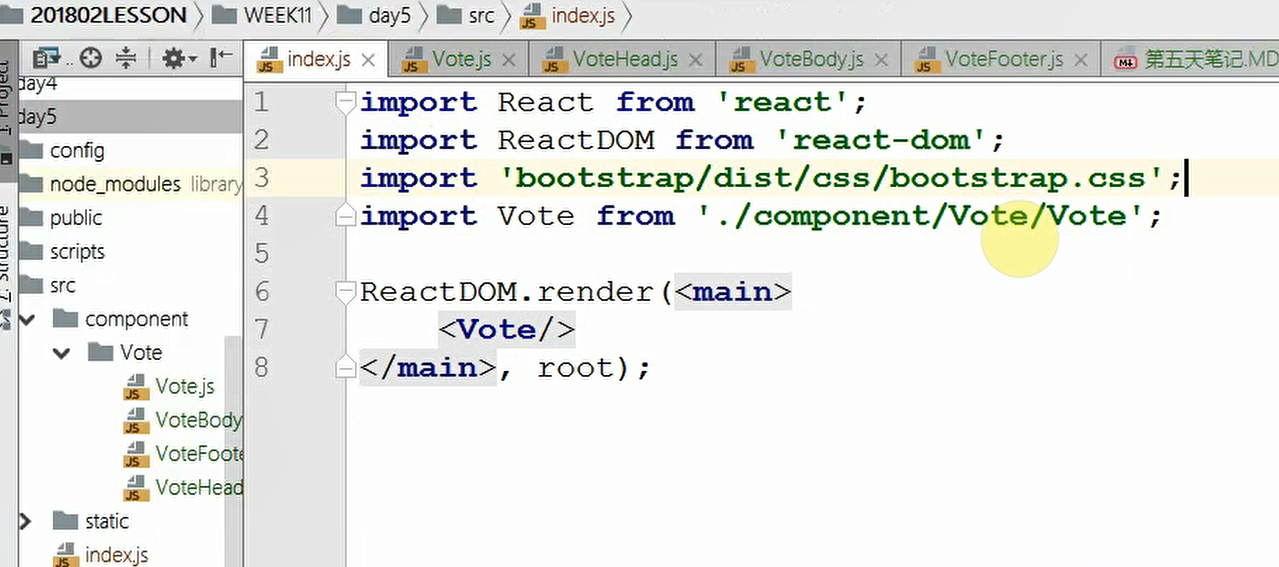
React进阶

# 组件包含组件的问题

1. 复合组件: 父组件嵌套子组件

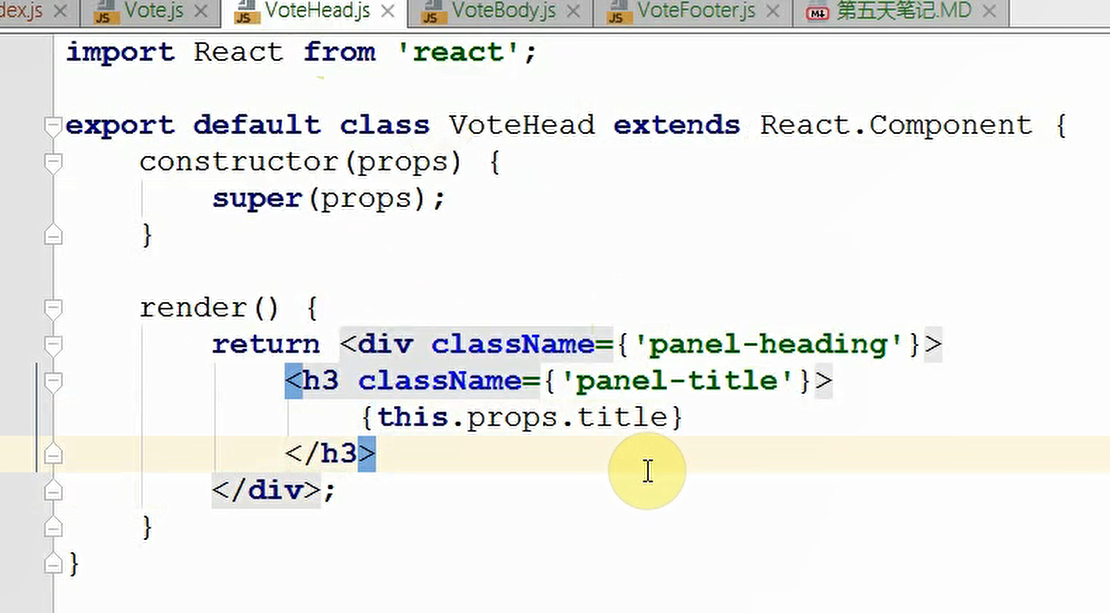


写个大组件.在大组件里面引用 小组件 .然后 在index.js中引入大组件

传递信息的方式:

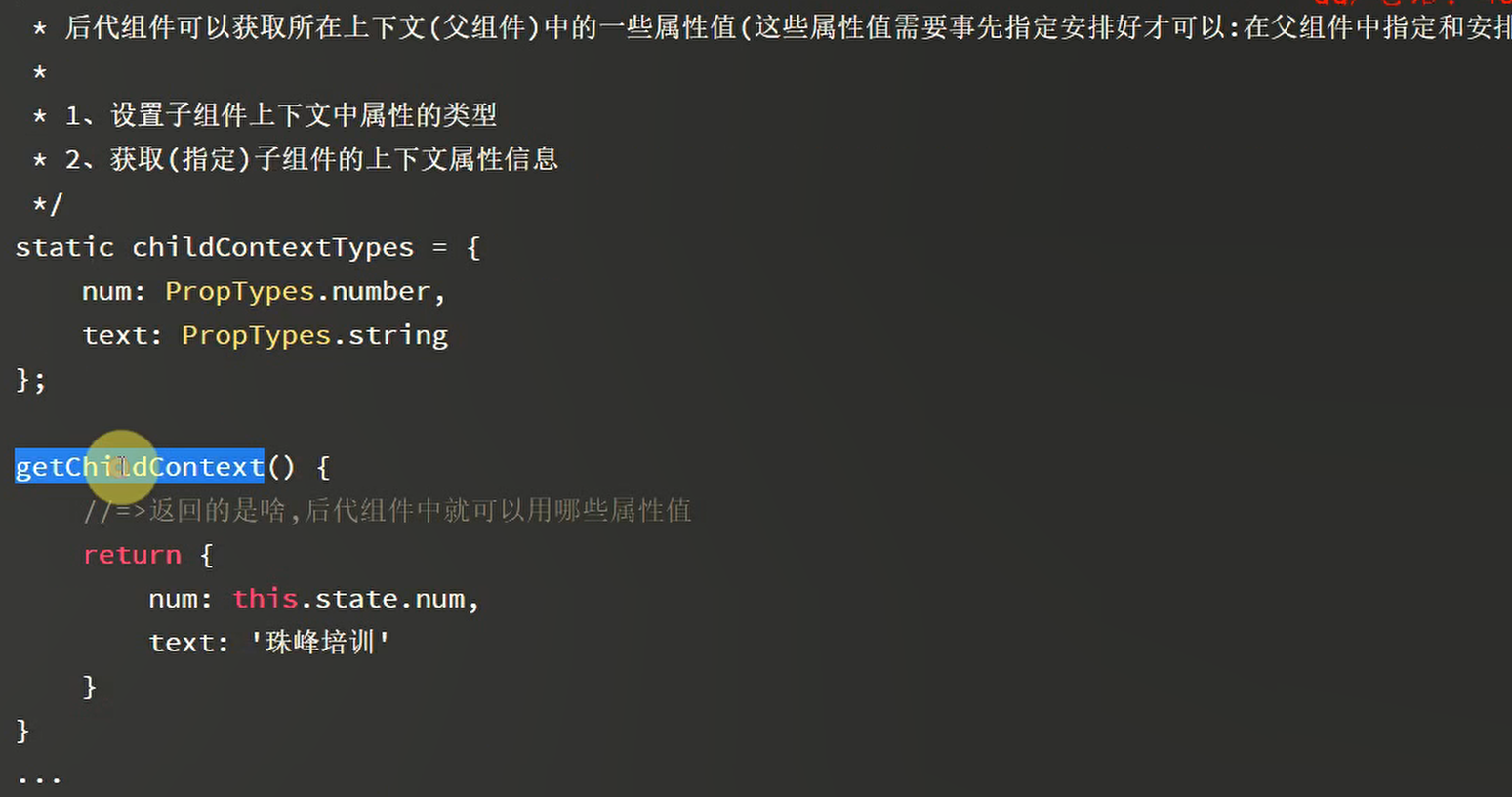
A: 父组件需要把信息传递给子组件

属性传递:调取子组件的时候,把信息基于属性的方式传递给子组件(子组件props存储传递信息).这种方式只能父传子,子组件无法直接把信息传递给父组件



在index.js中写 <vote > </vote>

上下文传递:



前提需要安装propTypes

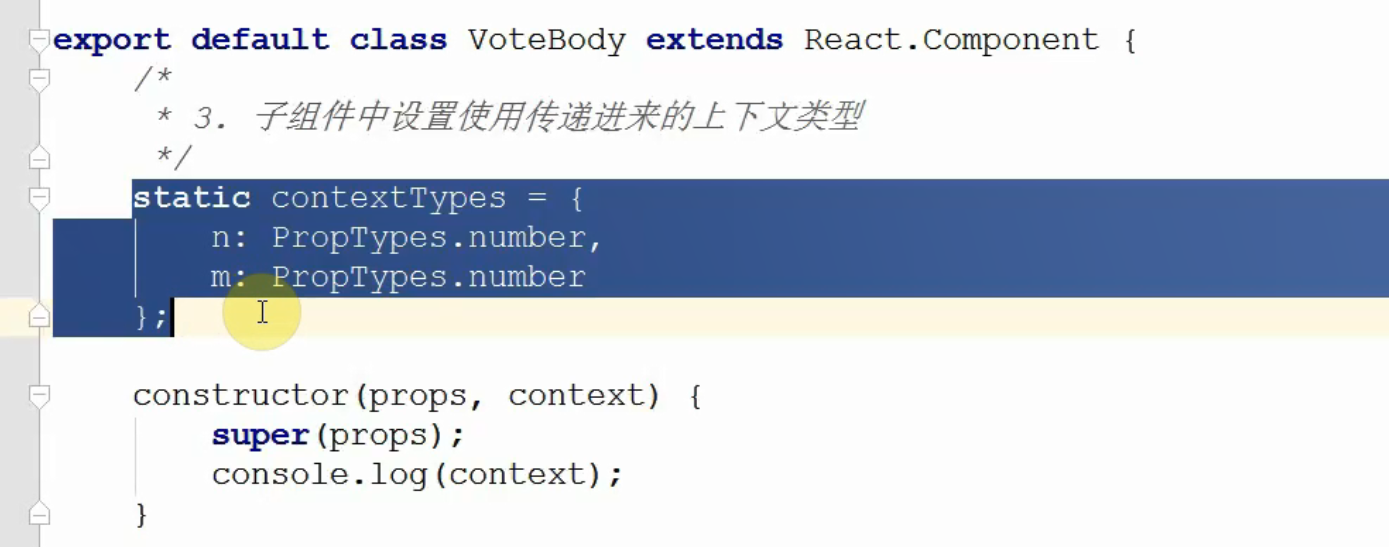
在父组件中

1. 设置上下文属性值类型’

static childContextTypes ={ }

1. 设置子组件的上下文属性值信息
2. getChildContext(){

return 的是啥.相当于给子组件上下文设置为啥}

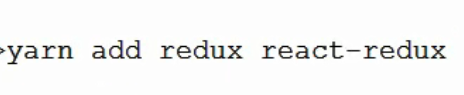


指定的上下文属性需要和父组件中指定的类型保持一致,否则报错

1. 平行组件: 兄弟组件或者毫无关系的组件

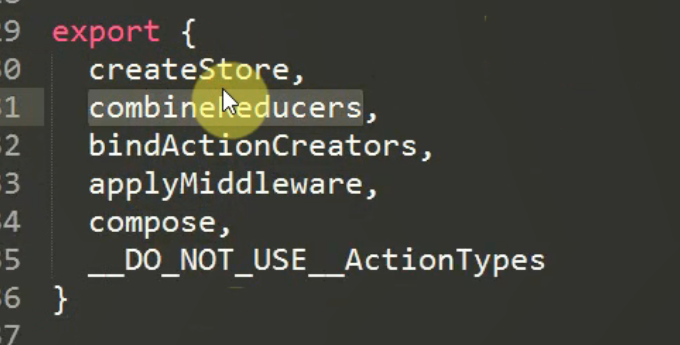
# Redux :

1. 安装



流程:





import {createStore } from ‘redux’;

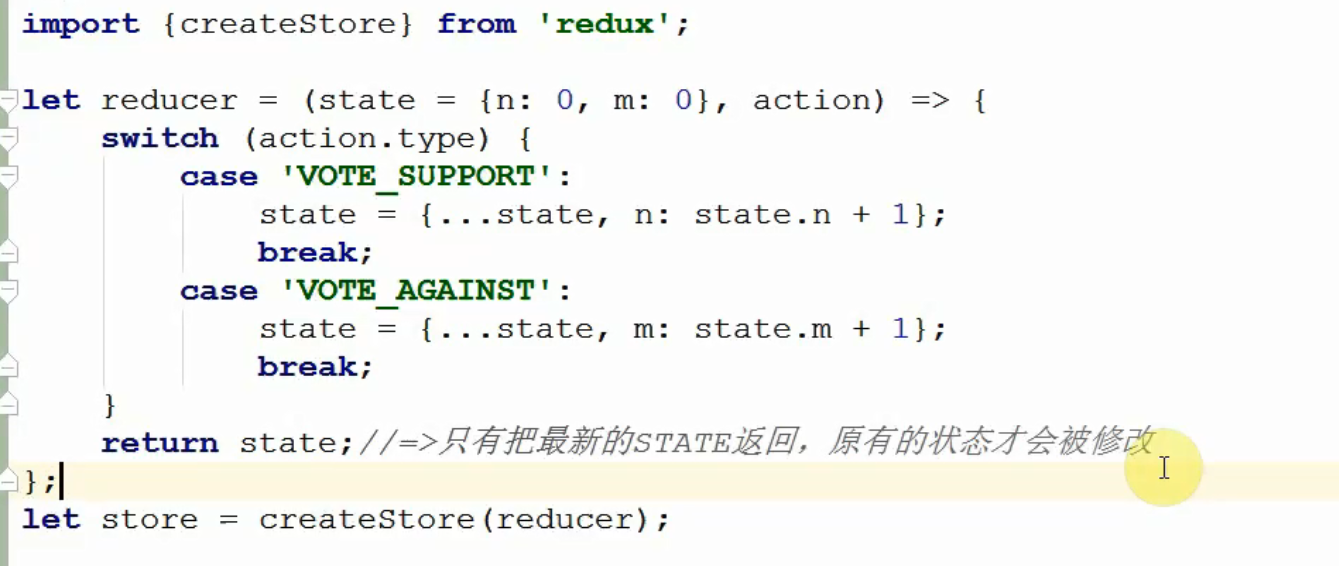
创建容器:需要把reducer传递进来( 登记了所有状态更新信息)

reducer作用:

1. 记录了所有状态修改的信息 (根据行为标识走不同的修改任务)
2. 修改了容器中的状态信息

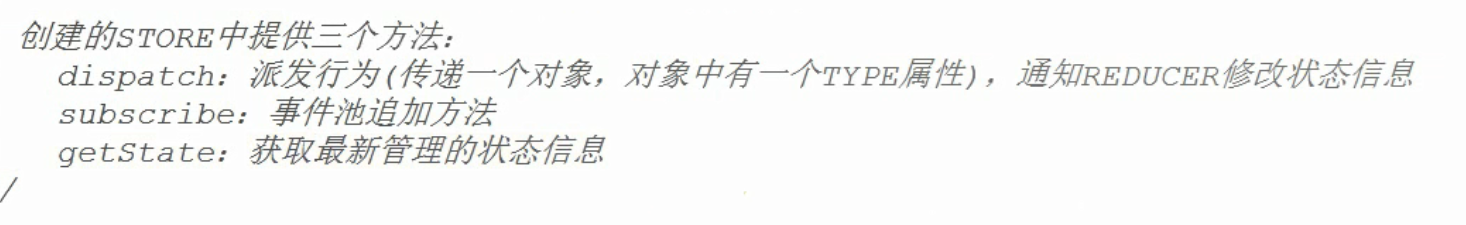
state:容器中原有的状态信息 (如果是第一次使用.没有原始状态.需要 设置一个初始默认值 )

action: dispatch任务派发的时候传递的行为对象(这个对象中必有一个type属性,是操作的标识,reducer就是根据这个行为标识来识别该如何修改状态信息)



创建好之后会返回三个方法 : 只有把最新的状态返回.原有的state才会修改

创建好的store中提供三个方法: action提供标识,就是给快递打个标签.根据这个标签类型(type)来修改.



---src---

|\_\_\_\_store { 创建商店的c.js {

首先要引入reducer.js

import {createStore} from ‘redux’

cost store =createStore( )

暴露出来:export default

|

| |

| |

| |

| |

|

reducer.js {

这是纪录本.纪录着一些信息.有哪些数据

export default ( store ,action) = > {return state} 整个图书的所有信息.store里的所有数据

const defaultState ={ input value=’’ list:1 , 2} 这个仓库默认有两个数据.由reducer管理着

|

|

|

|

|

|

|\_\_\_\_\_\_\_component   
 |

|\_\_\_\_\_\_\_todolist.js {

要引入 仓库 (你要用就要引入) import store from ‘./store/c.js’

创建类的方法创建组价

export default class Todolist extends React.Component {

constructor (props) {

super(props) ;

this.state= store.getSetstate()

}

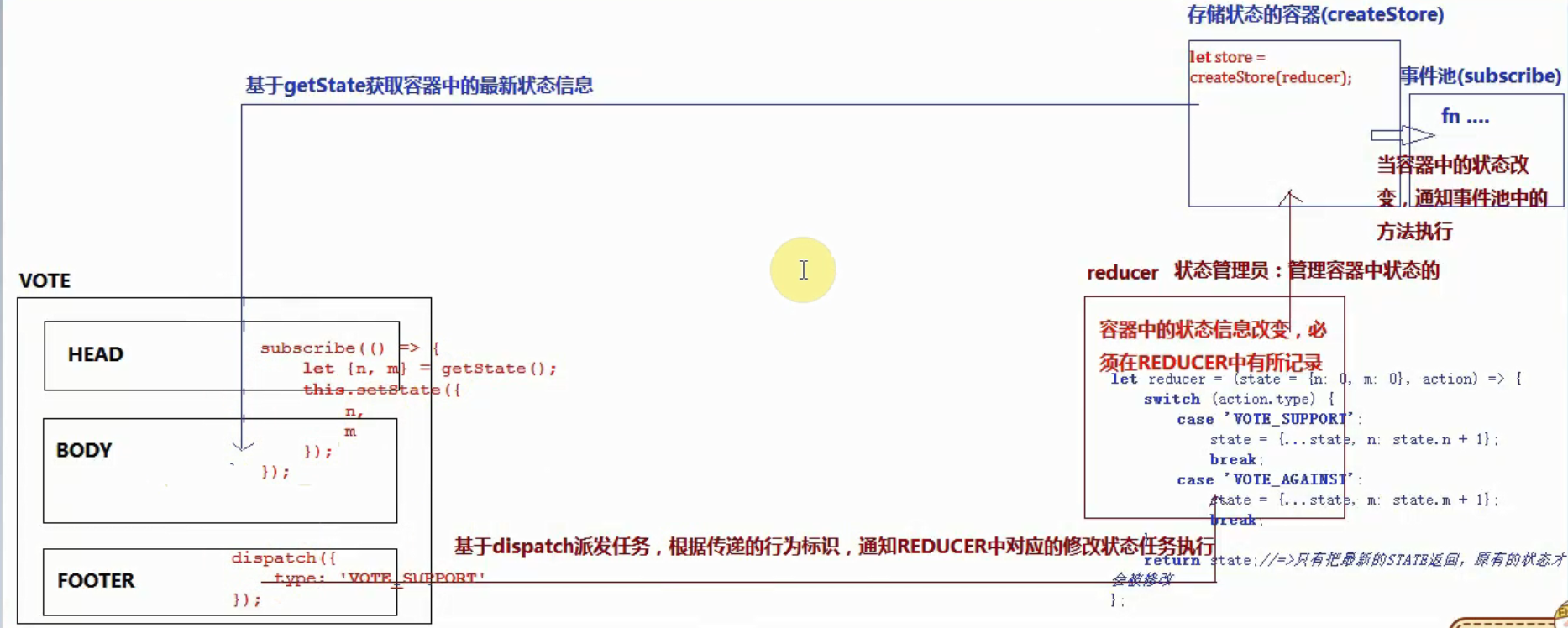
render () {

return }

}

<input value { this.state.inputvalue}

这里指的就是reducer里面的数据.用store.getState()获取使用.



### 工程项目:

store:

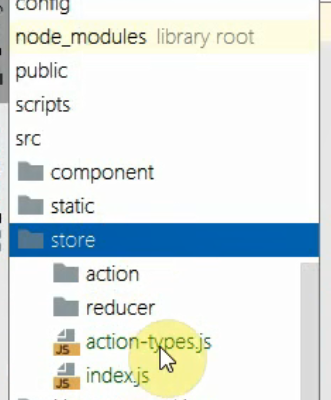
|

|\_\_\_ action

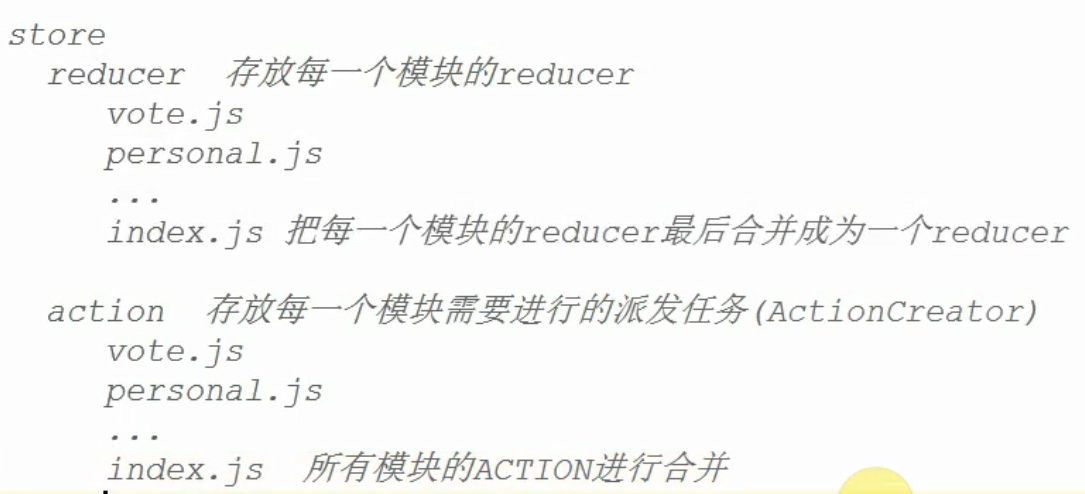
|\_\_\_\_reducer

|\_\_\_action-types.js

|\_\_\_\_index.js



#### 细节目录:



action-type.js :管控当前中所有REDUX任务派发中需要的行为标识

= > export const VOTE\_SUPPORT = ‘VOTE\_SUPPORT’

= > export const VOTE\_AGAINST = ‘VOTE\_ AGAINST’

## redux复习

what is redux?

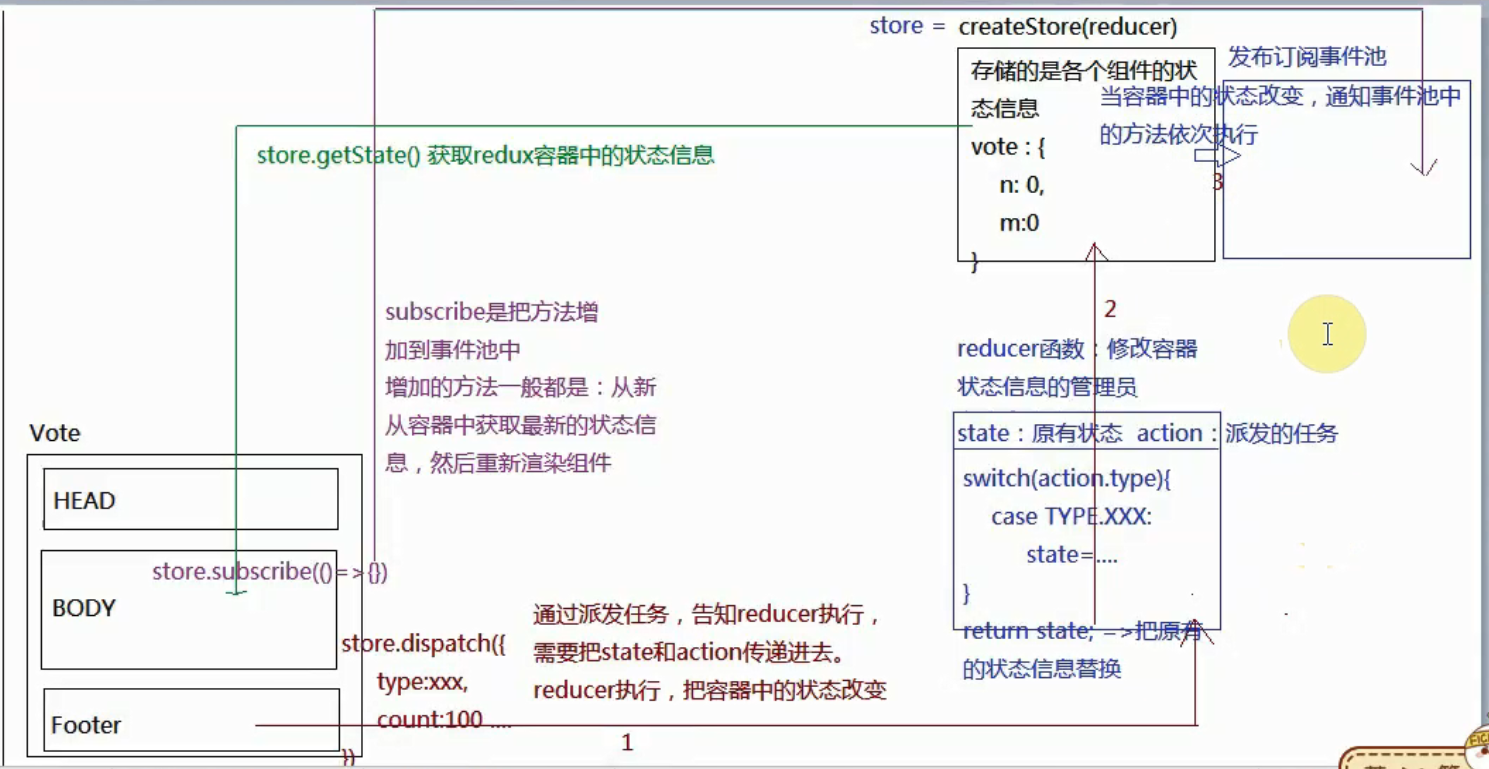
redux : 进行状态统一管理的类库. 适用于任何技术体系项目

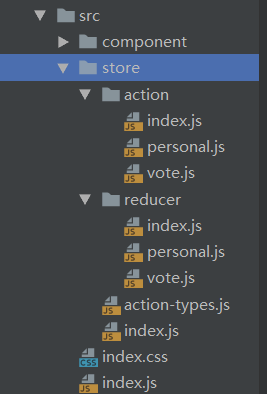
应用场景:

1. 只要两个或者多个组件之间想要实现信息的共享.都可以基于redux解决.把共享的信息存储到redux容器中进行管理
2. 使用redux做临时存储:页面加载的时候.把从服务器获取到的信息存储到redux中.直接从redux中拿取

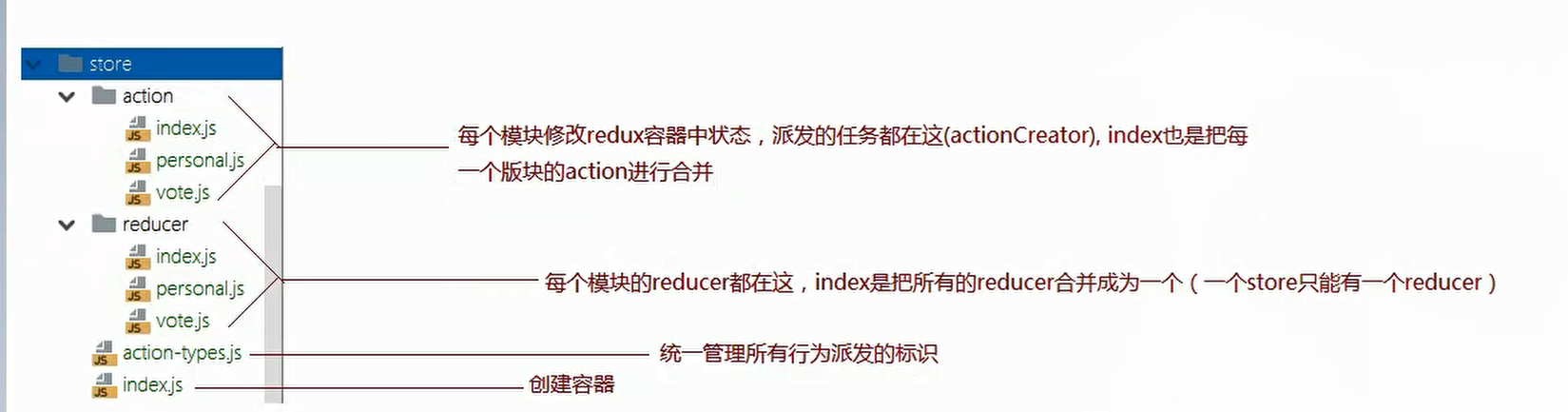


物流中心来理解.





redux各目录的关系

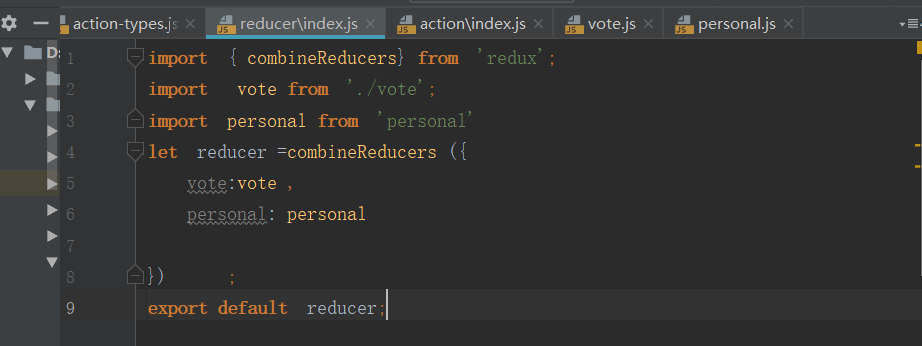

早reducer 的 personal.js中 ,纪录原始状态信息

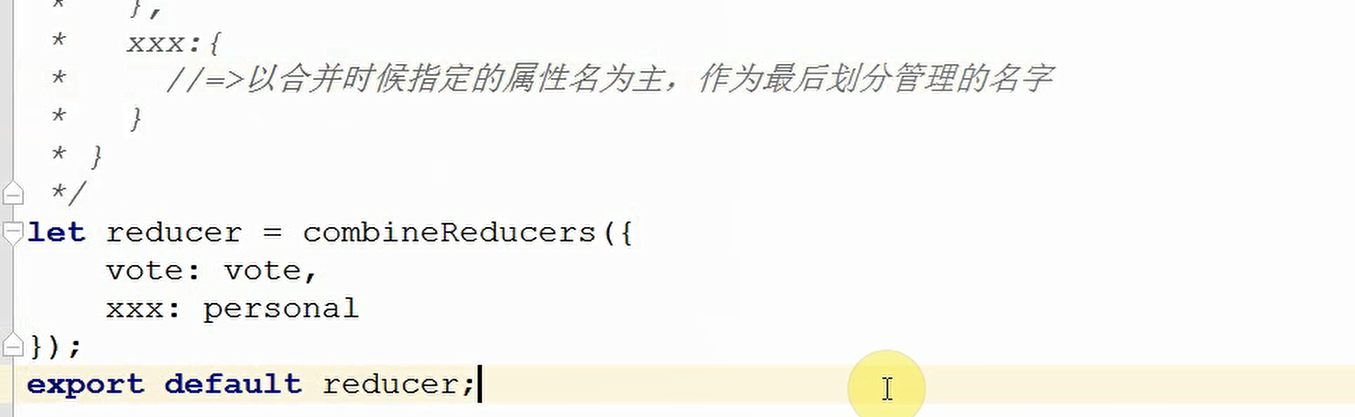
一般默认上述格式.

在reducer 的 index 中.合并各模块的状态纪录信息

使用combineReducer方法

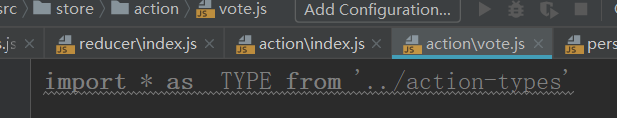
import { combineReducers } from ‘redux’ ( 要记得安装redux)





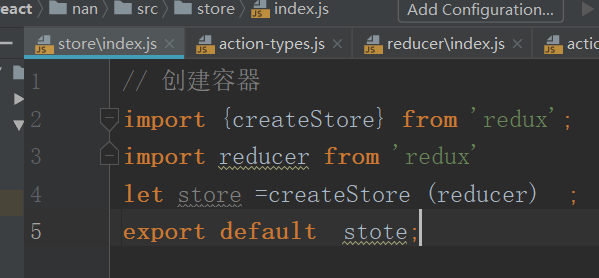
在action中的处理的事情暂时很无聊.就是封装几个方法

dispatch派发任务修改状态时候执行的方法.方法返回就是当前派发任务时候传递的action对象



引入其中.

在store 中的index 中创建容器



很繁杂

以上全部解决

着手解决 src 主目录下的 index

真是项目中我们会把redux容器中的状态信息获取到,赋值给组件的状态或者是属性(react-redux),这么做的目的是:当react的状态改变.我们可以把修改组件内部的状态,从而达到重新渲染组件的目的.

### react\_redux

react-redux是把redux进一步封装,适配react项目,让