# 这个练习的内容：

## 用react的语法实现下面的菜单效果，可以利用数组的map方法来实现

|  |
| --- |
|  |

### 有两个版本

### 注意：第一个菜单他没有子菜单，如果你直接用map方法会报错，处理方法1是使用三目运算符，处理方法2是在可能为空的对象上面加一个?

# 代码如下，

# 版本1：一级菜单之间有互斥性，一次只能展开一个一级菜单

## 我们还是在demo3项目里面写代码，新建一个sidemenu文件夹，在文件夹里面新建一个SideMenu.jsx组件，然后在App.jsx里面引用组件

### 项目结构

|  |
| --- |
|  |

### SideMenu.jsx

|  |
| --- |
| **import *React***, {Component} **from 'react'**;  **class** SideMenu **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,   **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ]  }   render() {  **let** {menus} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt**>{item.**title**}</**dt**>  {  *//写法1  // item.children? item.children.map(child=>{  // return(  // <dd key={child.id}>{child.title}</dd>  // )  //  // }):null  //写法2* item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   })  }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu; |

### 效果

|  |
| --- |
|  |

### 我们吧SideMenu.jsx的代码修改一下，添加一个openId属性，我们想实现的功能是选中的菜单会打开子菜单，没有选中的不会打开

|  |
| --- |
| **import *React***, {Component} **from 'react'**;  **class** SideMenu **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,   **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],  **openId**:0  }   render() {  **let** {menus} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt**>{item.**title**}</**dt**>  {  *//写法1  // item.children? item.children.map(child=>{  // return(  // <dd key={child.id}>{child.title}</dd>  // )  //  // }):null  //写法2* item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   })  }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu; |

### 我们利用这个openId来记录被点击的元素，在渲染二级菜单是时候需要判断 当前的id和openId是否相等，如果是就显示当前菜单对应的二级菜单，代码如下

### SideMenu.jsx

|  |
| --- |
| **import *React***, {Component} **from 'react'**;  **class** SideMenu **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,   **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],  **openId**:0  }   render() {  **let** {menus,openId} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt onClick=**{()=>{  **this**.setState({  **openId**: item.**id** })  }}>{item.**title**}</**dt**>  {  *//写法1  // item.children? item.children.map(child=>{  // return(  // <dd key={child.id}>{child.title}</dd>  // )  //  // }):null  //写法2* item.**id**===openId? item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   }):**null** }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu; |

## 老师的代码，先定义一个方法，然后在点击事件里面定义这个方法

|  |
| --- |
| **import *React***, {Component} **from 'react'**;  **class** SideMenu **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,   **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],  **openId**:0  }  *setOpenId*=(idx)=>{  **this**.setState({  **openId**:idx  })  }  render() {  **let** {menus,openId} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt onClick=**{()=>{  **this**.*setOpenId*(item.**id**)  }}>{item.**title**}</**dt**>  {  *//写法1  // item.children? item.children.map(child=>{  // return(  // <dd key={child.id}>{child.title}</dd>  // )  //  // }):null  //写法2* item.**id**===openId? item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   }):**null** }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu; |

## 注意：

### 1.修改openId的值需要使用setState方法

### 2.注意三目运算符的使用

### 效果

|  |
| --- |
|  |
|  |
|  |

## 假如我们需要一个功能，就是点击一下一级菜单会弹出二级菜单，再点击一次同一个一级菜单，会把二级菜单收回，怎么实现？

### 其实也好办，我们在事件处理函数里面先判断一下传入的openId和openId是否相等，如果说明他已经处于打开状态，我们只需要把openId设置为0即可，不相等才打开

|  |
| --- |
| **import *React***, {Component} **from 'react'**;  **class** SideMenu **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,   **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],  **openId**:0  }  *setOpenId*=(openId)=>{  **if**(openId == **this**.**state**.**openId**){  **this**.setState({  **openId**:0  })  } **else** {  **this**.setState({  openId  })  }   }  render() {  **let** {menus,openId} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt onClick=**{()=>{  **this**.*setOpenId*(item.**id**)  }}>{item.**title**}</**dt**>  {  *//写法1  // item.children? item.children.map(child=>{  // return(  // <dd key={child.id}>{child.title}</dd>  // )  //  // }):null  //写法2* item.**id**===openId? item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   }):**null** }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu; |

### 注意这里有一个简写：当函数的形参和属性名一样时可以简写，参考render()上面的else代码段

### 版本一完成

# 版本2:这个版本可以同时展开多个菜单，菜单之间没有互斥新

## 我们在src目录下面新建一个side-menu-v2文件夹在里面新建一个SideMenu2.jsx,用快捷键rcc创建类组件骨架

|  |
| --- |
| //SideMenu2.jsx  **import *React***, {Component,PureComponent} **from 'react'**;  **class** SideMenu2 **extends** PureComponent {  render() {  **return** (  <**div**>   </**div**>  );  } }  **export default** SideMenu2; |

## 我们先把SideMenu.jsx的代码复制过来然后把一些不需要的代码删除，变为下面的样子

|  |
| --- |
| **import *React***, {Component,PureComponent} **from 'react'**;  **class** SideMenu2 **extends** PureComponent {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,   **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],   }   render() {  **let** {menus} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt**>{item.**title**}</**dt**>  {  item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   })  }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu2; |

## 为了实现上面的功能我们需要给有子菜单的列表添加一个open属性，然后使用在一级菜单的点击事件里面把open属性取反，调用setState方法，然后进行条件渲染

|  |
| --- |
| **import *React***, {Component,PureComponent} **from 'react'**;  **class** SideMenu2 **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **open**:**true**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,  **open**:**true**,  **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],   }   render() {  **let** {menus} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt onClick=**{()=>{  item.**open** = !item.**open** *// alert(item.open)* **this**.setState({  ...**this**.**state** })  }}>{item.**title**}</**dt**>  {  item.**open**? item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   }):**null** }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu2; |

### 我的版本2，定义一个setOpen的方法

|  |
| --- |
| **import *React***, {Component,PureComponent} **from 'react'**;  **class** SideMenu2 **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **open**:**true**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,  **open**:**true**,  **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],   }  *setOpen* = (id)=>{  **this**.**state**.**menus**.forEach(item=>{  **if**(item.**id** ===id){  item.**open** = !item.**open** }  })  **this**.setState({  ...**this**.**state** })  }  render() {  **let** {menus} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt onClick=**{()=>**this**.*setOpen*(item.**id**)}>{item.**title**}</**dt**>  {  item.**open**? item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   }):**null** }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu2; |

### 老师的版本。还是定义一个叫做setOpen的方法

|  |
| --- |
| **import *React***, {Component,PureComponent} **from 'react'**;  **class** SideMenu2 **extends** Component {  **state**={  **menus**:[  {**id**:1,**title**:**'首页'**},  {  **id**:2,**title**:**'商品管理'**,  **open**:**true**,  **children**:[  {  **id**:**'2-1'**,**title**:**'商品列表'** },  {  **id**:**'2-2'**,**title**:**'新增商品'** }   ]  },  {  **id**:3,**title**:**'订单管理'**,  **open**:**true**,  **children**: [  {  **id**:**'3-1'**,**title**:**'订单列表'** },  {  **id**:**'3-2'**,**title**:**'新增订单'** }  ]  }  ],   }  *// setOpen = (id)=>{  // this.state.menus.forEach(item=>{  // if(item.id ===id){  // item.open = !item.open  // }  //  // })  // this.setState({  // ...this.state  // })  // }  setOpen* = (id)=>{   **this**.setState({  **menus**:**this**.**state**.**menus**.map(item=>{  **if**(item.**id** ===id){  **return** {  ...item,*//保留对象旧属性* **open**:!item.**open** }  }  **return** item  })  })  }  render() {  **let** {menus} = **this**.**state  return** (  <**div**>  {  menus.map(item=>{  **return** (  <**dl key=**{item.**id**}>  <**dt onClick=**{()=>**this**.*setOpen*(item.**id**)}>{item.**title**}</**dt**>  {  item.**open**? item.**children**?.map(child=>{  **return**(  <**dd key=**{child.**id**}>{child.**title**}</**dd**>  )   }):**null** }    </**dl**>  )  })  }  </**div**>  );  } }  **export default** SideMenu2; |

### 效果:不管其他一级菜单有没有展开，只要没有展开的有二级菜单的一级菜单，点击一下会展开，再点击一下会折叠，可以展开多个菜单

# 总结：

## 在react修改数据一定要使用setState方法，虽然我们只修改一个属性，但是还是必须修改整个state，最好使用数组的map方法，因为他可以返回一个新数组，然后我们把新数组代替原来的数组。注意对象的处理方法，使用...来保留原来的属性，然后重新设置需要修改的属性。