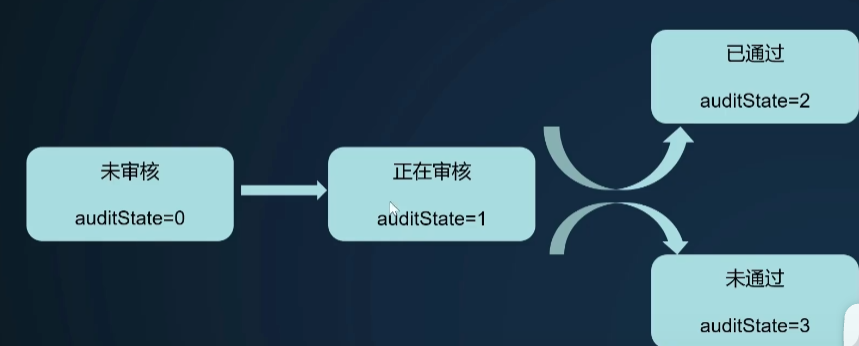
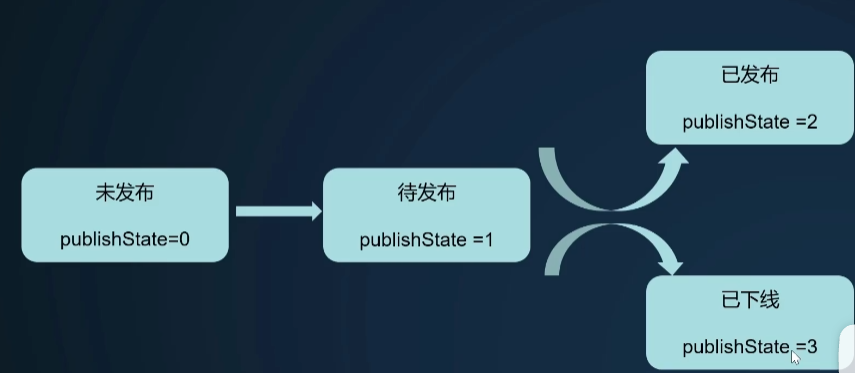
### 审核流程



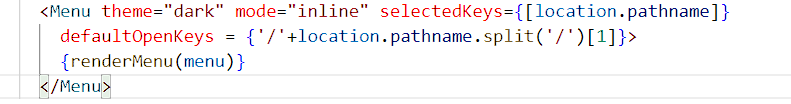
### 发布流程



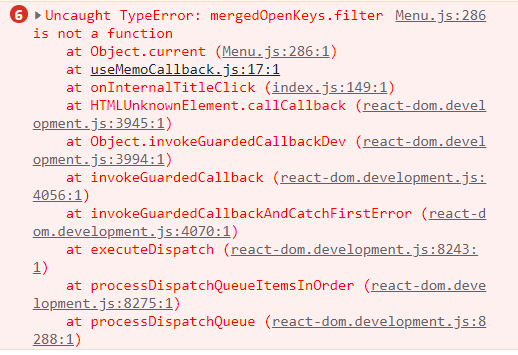
### 一、侧边栏

1、判断侧边栏一级目录是否可折叠（选中），利用item.children?.length

2、页面刷新或路由到其他页面，侧边栏直接重置了而不是留在当前路由页面选项高亮，获取当前路由。利用const location = useLocation();后的location.pathname获取当前URL设置到Menu的selectKeys属性上，而设置其defaultOpenKeys属性值为一级路由名从而控制一级列表的展开



3、3月6日，侧边栏突然出现bug，二级菜单不渲染，查找原因后发现defaultOpenKeys报错，location.pathname在加载时输出正确url而加载完毕后输出一个’/’。



### 二、权限列表

1、删除某一权限时，若是一级权限（item.grade === 1）则设置状态为过滤掉将要删除状态的一项setdataSource(dataSource.filter(data => data.id !== item.id))。若是二级权限，先过滤出有相同一级权限的二级权限形成列表然后再设置状态为过滤掉将要删除状态的一项。

2、设置自身某一权限时，直接更改权限的permission值，拥有权限为1，设置关闭为0，直接更改状态并利用patch更新后台。

3、删除或改变某一权限状态的时候，侧边栏不是实时更新，采用window.location.reload()

方法强制刷新界面（有很大待优化空间）

### 三、角色列表

1、roles接口中并没有key属性，利用Table组件时如果没找到key可以使用rowKey属性指定一个函数的返回值作为key值rowKey={(item)=>item.id}

2、

### 四、UserList

1、内置超级管理员admin设置其用户状态和操作为不可修改

2、使用useEffect钩子时一定要传入第二个参数（大部分为空数组[]），否则页面无论会不会变化都会一直调用axios请求，会卡死。

3、通过在子组件中箭头函数括号前面套forwardRef进行ref的透传，外层的Ref能绑定到里层上面

4、ref.current.setFieldsValue({region:""})可以设置父组件传过来的props中的值，如例子中的region

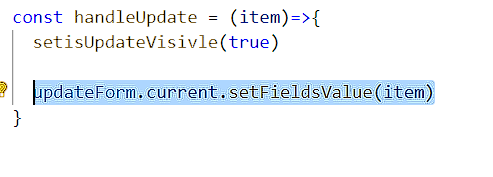
5、新增一个用户时，若是超级管理员则地区项不是必填的，利用rules={isDisable?[]:

[{ required: true, message: '请选择区域！' }]}进行设置

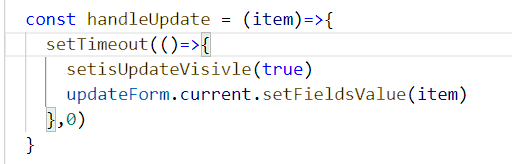
5、新增一个用户点确定时，首先要post到后端以生成id再设置dataSource，方便后续的删 除和更新操作，setdataSource时直接setdataSource([...value],res.data)会导致页面的新建 角色名称为空需要：



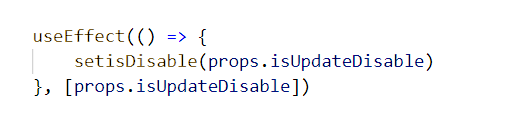
6、更新用户信息时，在按钮事件中



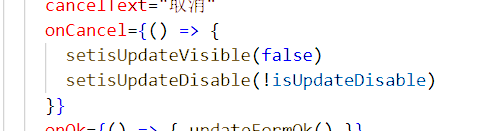
第一次好用第二次会报错。因为更新状态不是同步的，必须等表单内容创建完毕之后再调用setFieldsValue，可以把他们俩改为同步操作，套入一个setTimeout中。



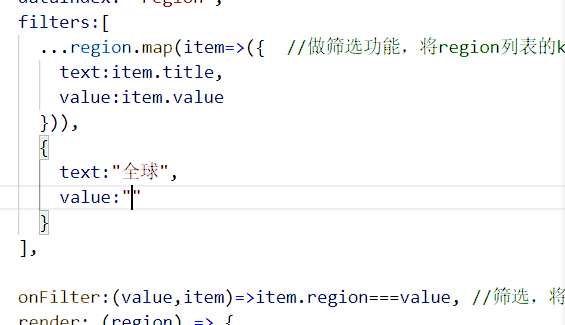
7、更新用户信息时，若用户是超级管理员，要将区域属性设置为不可改变，UserList中判断item.grade===1则为超级管理员，设置一个禁改状态，并利用props传给UserForm，在其中



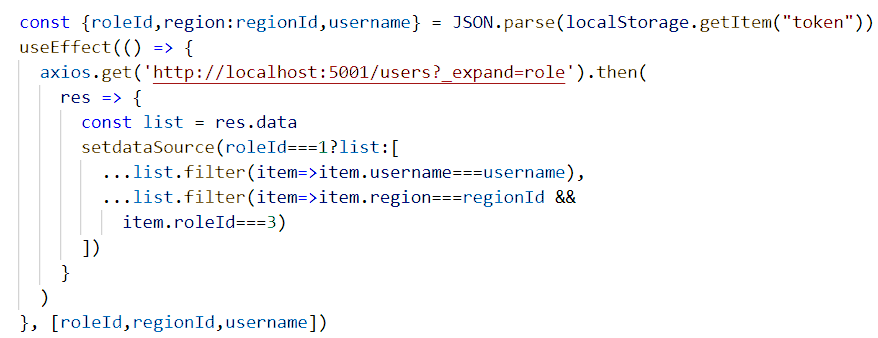
8、更新用户信息时，若选择角色后取消不改变，他自己的禁改状态没问题不过会影响别人，所以每次取消时，禁改状态取反



9、做筛选功能，将region列表的key值转换为antd table筛选所需的text和value字段，将所有region为value的item显示出来。全球region的value值设置为“”



10、登陆后判断roleId等级进行筛选，若是超级管理员直接后端全部请求显示用户，否则只把自己和自己所在地区的所有区域编辑显示出来。



11、登陆后使用UserForm时（新增用户和修改用户）在UserForm中判断权限从而限制其区域和权限修改程度。



### 五、登录

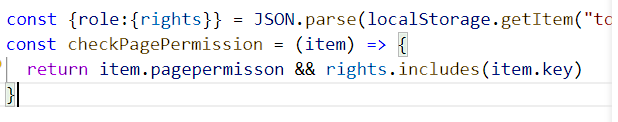
1、退出登陆前要清除缓存localStorage.removeItem("token")

2、登录时除了判断username和password之外还要判断roleState是否为true（是否被admin封号了）。还需要\_expand=role联表查询角色信息和权限。

3、登录操作进行axios的get请求时，因为$前面没有/，使用模板字符串请求会产生乱码，使用字符串拼接方式完成请求



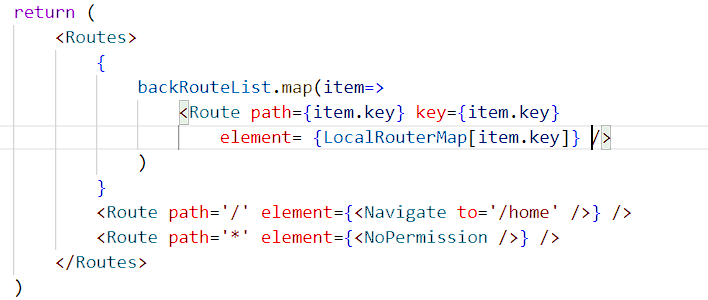
4、登陆后的侧边栏，先从localStorage中取该用户的权限然后渲染时判断是否显示出来。



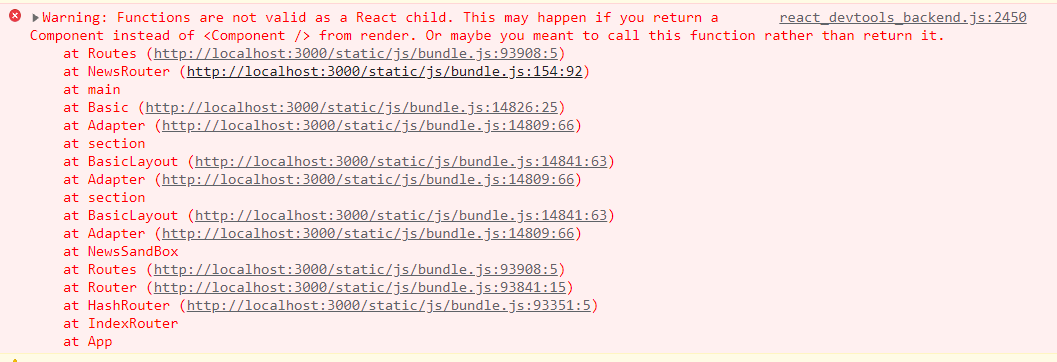
### 六、路由权限认证

1、动态创建路由时





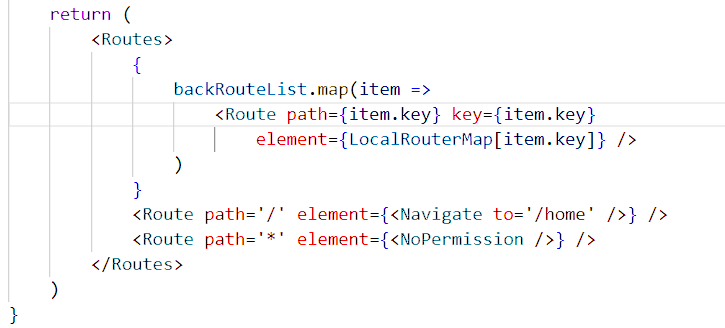
会发生错误：



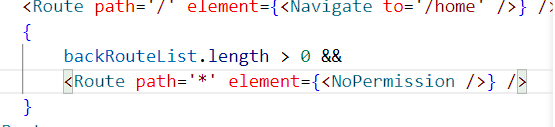
原来是配置路由的时候Map值设为模块而不是标签，改成下面就好了



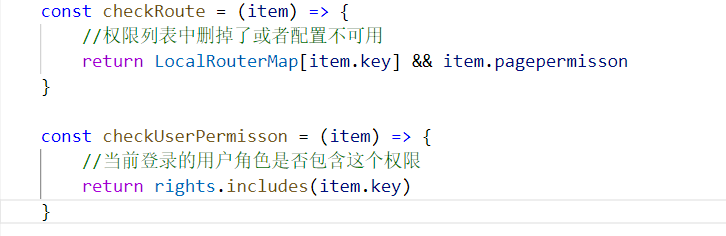
2、刷新后瞬间会出现403界面，因为最开始backRouteList是空数组，不渲染，而会渲染下面的403



把403的路由做判断，只有当backRouteList不是空数组还匹配不到路由时才显示403



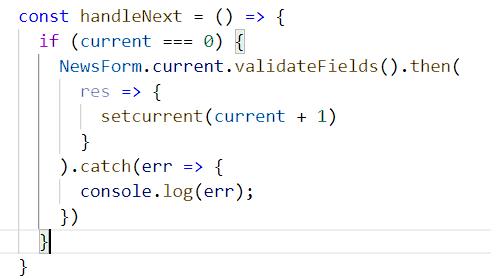
3、动态设置路由的时候，要分别判断权限列表中删掉了或者配置不可用和当前登录的用户角色是否包含这个权限的情况。这样的话即使知道路由地址也不能通过url直接访问，直接实现限制权限



### 七、撰写新闻：

1、审核条步骤又状态current控制，但不要根据current的值切换不同步骤显示的div，在三个div中增加hidden属性即可，这样回到上一步骤的时候div中存储着输入过的数据

2、利用ref={NewsForm}判断撰写新闻的时候验证是否通过（form自带的submit button被自定义的下一步button代替）。

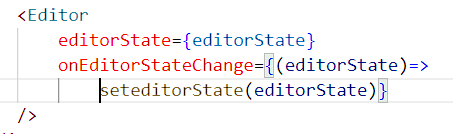


3、富文本编辑器，利用

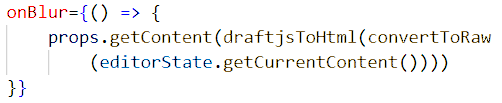
import { Editor } from 'react-draft-wysiwyg'

import 'react-draft-wysiwyg/dist/react-draft-wysiwyg.css'

将其变为受控组件，更新的时候，只要传过来一个新的editorState会配合新的显示信息

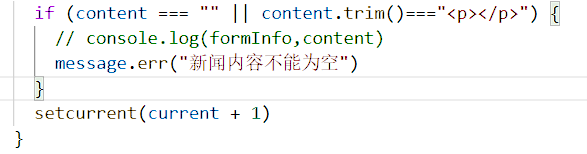


4、将富文本编辑器内容转化为html片段



利用props函数传回NewsAdd组件新闻主体内容

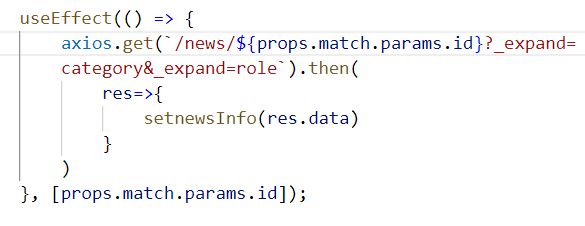
5、新闻内容若为空或者输入过又删除了，则提示新闻内容不能为空，不为空则进如下一步骤。



### 八、草稿箱

1、动态配置路由时，每一个新闻都有一个routepermisson属性用来判断当前用户是否有资格创建预览改新闻的路由，每一行的新闻标题设置为超链接标签，根据点击的新闻id跳转到对应预览路由。

2、在预览界面中获取当前路由页面的参数id，向后端请求时要把新闻相关信息都请求回来，包括作者信息和发布区域。



3、状态为

直接在动态渲染的时候使用

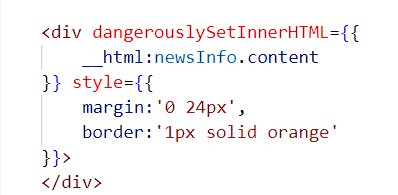
此时异步请求ajax还没返回呢，直接null.title会报错之后才返回数据。

解决：或者渲染之前判断newsInfo是否返回数据了

4、用模板字符串写url的时候不能写在多行上否则无法访问获取数据！使用navigate跳转路由页面的时候前面一定要加上‘/’，否则会出现如下错误



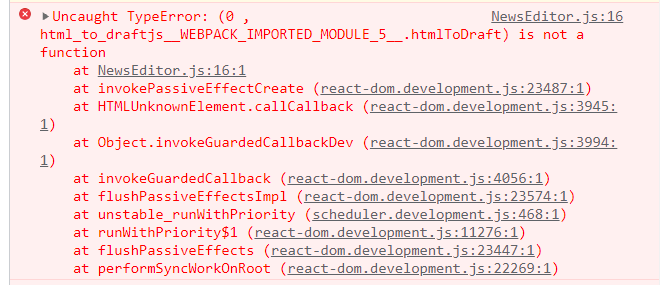
5、利用富文本编辑器存入后台的新闻主体内容为HTML语言内容，预览时要转化回去



6、更新新闻时，要把原新闻内容设置到更新位置，给Form设置ref={NewsForm}后，请求回数据之后，设置数据。



7、将后台新闻主体的html转化为draft时，使用htmlToDraft报错



### 九、审核列表

1、显示新闻筛选条件：作者是自己，auditState ！== 0（0是草稿箱中的），publishState ！== 2||3（2是已发布，3是已下线）

2、json-server中发送请求时，\_ne是不等于，\_lte是小于等于

### 十、审核

1、审核页面要区分取回的新闻：审核状态是待审核

### 十一、新闻分类

1、可编辑单元格：

A、给table添加属性

components={{

          body: {

            row: EditableRow,

            cell: EditableCell,

          }

        }}

B、把EditableCell和EditableRow抄过来

C、给需要可编辑的列增加（handleSave是失去焦点的回调）

onCell: (record) => ({

        record,

        editable: true,

        dataIndex: "title",

        title: "栏目名称",

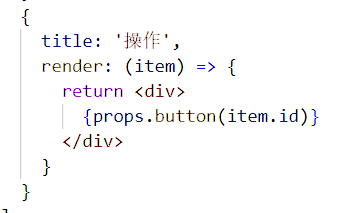
        handleSave: handleSave,

      })

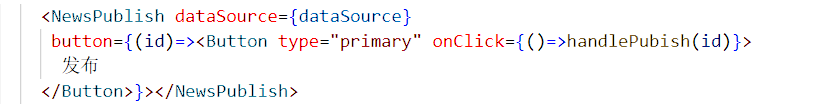
### 十二、发布管理

1、因为代码复用性太高，自定义hooks，传入发布状态获取数据

2、将table单独封装成一个组件，但是当按钮被点击时，需要获得当前节点，在table组件中“操作”栏的render调用时，传入被点击当前节点所代表的新闻id

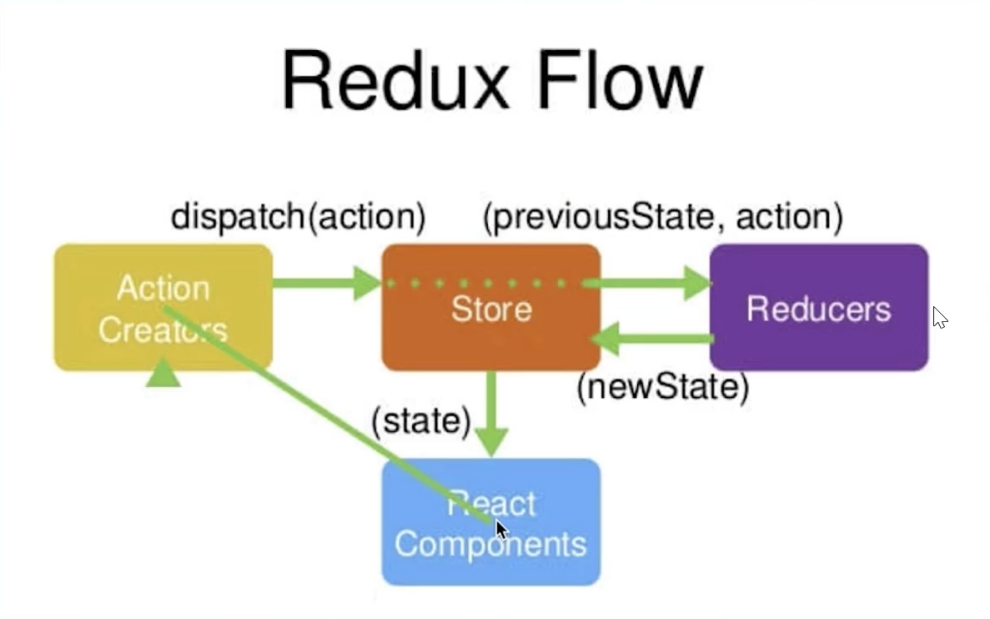


在已发布等列表中接收到id发给hooks中定义函数处理（！注意button写法）



### 十三、Redux

1、侧边栏与小按钮身处兄弟组件，交互状态麻烦，引入redux

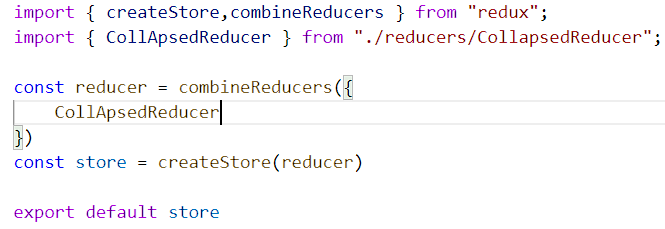




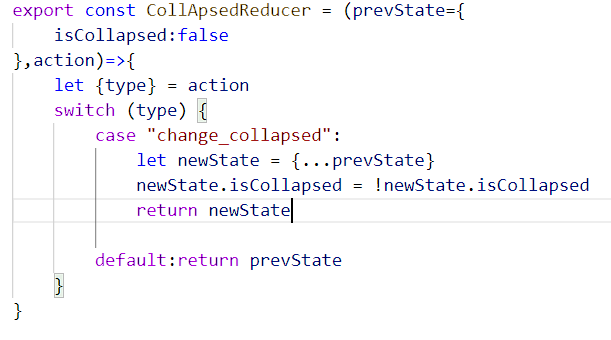
简单来说为了得到状态值，为了修改状态值

使用时用。

Store.js中：



CollApsedReducer.js中：



### 十四、loading功能（axios拦截）

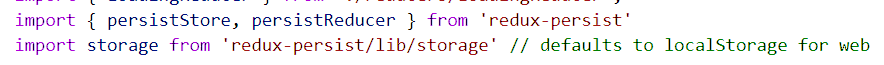
1、loading包在sandbox里动态生成路由的外面。

2、axios拦截器，每次请求之前必然走

请求结束后必然走

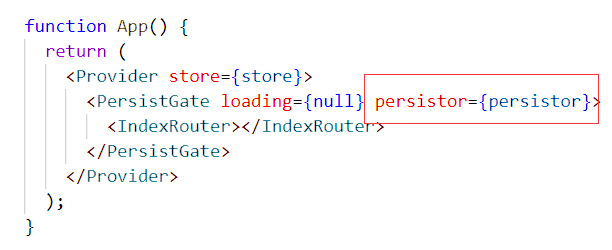
### 十五、持久化状态

1、在store中配置





在app中配置



### 十六、首页

1、用户最常浏览：

axios.get(`/news?publishState=2&\_expand=category&\_sort=view&\_order=desc&\_limit=6`)

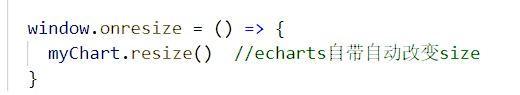
发布状态为2，得到新闻类别，排序，倒序，拿前六个

2、可视化图标（柱状图）：echarts 

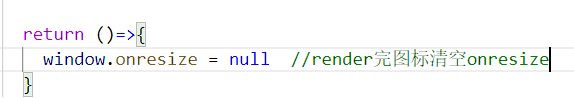
3、将返回的新闻根据类别进行分类，引入 



4、渲染完柱状图，当窗口改变的时候柱状图也要随之改变：（echarts自带resize方法）



在useEffect之后要清空onresize



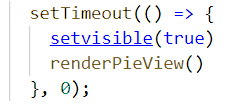
5、渲染饼状图时，若直接在按钮中写



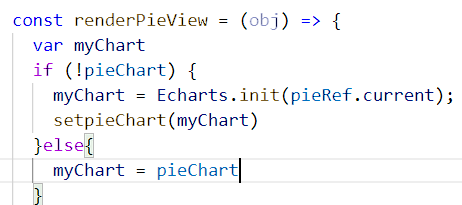
会报错



这是因为饼状图还没创建就设置它的可见值，解决方法，套入定时器中变成异步操作即使设置的时间为0：



6、直接把初始化放入生成饼状图函数中会初始化多次浪费资源，设置状态判断是否初始化过：



### 十七、游客系统

1、得到全部新闻信息后用-分组



得到的数据结构如下：



为了能在list显示需要转化为新的数据结构（二维数组）：

