**const** DEFAULT\_QUERY='redux';

**const** PATH\_BASE='https://hn.algolia.com/api/v1';  
**const** PATH\_SEARCH='/search';  
**const** PARAM\_SEARCH='query=';  
**const** PARAM\_PAGE='page=';  
**class** App2 **extends** Component  
{  
 constructor(props)  
 {  
 //console.log(url);  
 **super**(props);  
 **this**.state = {  
 searchItem: DEFAULT\_QUERY,  
 searchKey:'',  
 results: **null**,  
 error:**null** };  
 //点击删除的按钮的类的方法  
 **this**.onMiss = **this**.onMiss.bind(**this**);  
 //搜索的类的方法  
 **this**.onSearch = **this**.onSearch.bind(**this**);  
 //搜索后显示的方法  
 // this.isSearch = this.isSearch.bind(this);  
 //绑定捕获结果的方法  
 **this**.fetchSearchTopStories=**this**.fetchSearchTopStories.bind(**this**);  
 //在服务端进行搜索的方法  
 **this**.onSearchSubmint=**this**.onSearchSubmint.bind(**this**);  
 //处理获取到的数据的方法；  
 **this**.setSearchTopStories=**this**.setSearchTopStories.bind(**this**);  
 //绑定当客户端缓存有要去搜索的东西的时候阻止API搜索的方法  
 **this**.needsToSearchTopStories=**this**.needsToSearchTopStories.bind(**this**);  
 };  
 //处理获取到的数据  
 setSearchTopStories(data,page)  
 {  
 //分页抓取  
 **const** {results,searchKey}=**this**.state;  
 //results已经存在并且已经有了该键值的子数组  
 **const** oldList = results&&results[searchKey]?results[searchKey].hits:[];  
 **const** updateList = [...oldList,...data.hits];  
 **this**.setState({  
 results:{  
 ...results,  
 [searchKey]:{hits:updateList,page:page}  
 },  
 });  
 }  
 //获取数据的方法  
 fetchSearchTopStories(searchItem,page=0)  
 {  
 fetch(`${PATH\_BASE}${PATH\_SEARCH}?${PARAM\_SEARCH}${searchItem}&${PARAM\_PAGE}${page}`)  
 .then(response => response.json())  
 .then(data =>**this**.setSearchTopStories(data,page))  
 //错误处理：  
 .catch(e=>**this**.setState({error:e}));  
 };  
 //利用生命周期完成渲染的时候去渲染获取  
 componentDidMount()  
 {  
 **const** {searchItem}=**this**.state;  
 **this**.setState({  
 searchKey:searchItem  
 });  
 **this**.fetchSearchTopStories(searchItem);  
 }  
  
 //搜索的方法  
 onSearch(event){  
 **this**.setState({  
 searchItem:event.target.value  
 })  
 }  
  
 //搜索后显示的类的方法---高阶函数  
 /\*isSearch(id)  
 {  
 return (item)=>item.title.toLowerCase().includes(id.toLowerCase());  
 }\*/  
  
 //删除后重新渲染  
 onMiss(id)  
 {  
 **const** {results,searchKey}=**this**.state;  
 **const** {hits,page}=results[searchKey];  
 **const** isNot=(item)=> item.objectID!==id;  
 **const** updateHits=hits.filter(isNot);  
 **this**.setState({  
 results:{  
 ...results,  
 [searchKey]: {hits: updateHits,page}  
 }  
 })  
 }  
 //定义阻止的方法  
 needsToSearchTopStories(searchItem)  
 {  
 **return** !**this**.state.results[searchItem];  
 }  
 //在服务端进行搜索的方法的定义(如果已经存在客户端的缓存区就停止搜索的API)  
 onSearchSubmint(event)  
 {  
 **const** {searchItem}=**this**.state;  
 **this**.setState({searchKey:searchItem});  
 **if**(**this**.needsToSearchTopStories(searchItem)){**this**.fetchSearchTopStories(searchItem);}  
 //防止源生js点击submint事件后重新渲染  
 event.preventDefault();  
 }  
 render() {  
 **const** {results, searchItem,searchKey,error} = **this**.state;  
 **const** page=(  
 results&&results[searchKey]&&results[searchKey].page  
 )||0;  
 **const** list=(  
 results&&results[searchKey]&&results[searchKey].hits  
 )||[];  
 /\*判断获取啥数据  
 console.log(searchKey);\*/  
 //错误处理  
 **if** (error){ **return** <p>获取数据失败</p> }  
 **return** (  
 <div className='page'>  
 <Search searchItem={searchItem} onSearch={**this**.onSearch} onSubmint={**this**.onSearchSubmint}>  
 搜索：  
 </Search>  
 <Table  
 list={list}  
 /\*isSearch={this.isSearch}  
 pattern={searchItem}\*/  
 onMiss={**this**.onMiss}>  
 </Table>  
 <Button onClick={()=>**this**.fetchSearchTopStories(searchKey,page+1)}>获取更多</Button>  
 </div>  
 )  
 }  
}  
//Search组建  
**class** Search **extends** Component  
{  
 render()  
 {  
 **const** {onSearch,searchItem,children,onSubmint}=**this**.props;  
 **return**(  
 <div>  
 <form onSubmit={onSubmint}>  
 {children} <input type='text'  
 onChange={onSearch}  
 value={searchItem}  
 />  
 <button type='submint'>{children}</button>  
 </form>  
 </div>  
 )  
 }  
}  
//Table组件  
**class** Table **extends** Component  
{  
 render()  
 {  
 console.log(**this**.props.list);  
 **const** {list,onMiss}=**this**.props;  
 **return**(  
 <div className='table'>  
 {list.map(item=>  
 <div key={item.objectID}>  
 <span>  
 标题： <a href={item.url}>{item.title}</a>  
 </span><br/>  
 作者：<span>{item.author}</span><br/>  
 评论：<span>{item.num\_comments}</span><br/>  
 观点：<span>{item.points}</span>  
 <Button onClick={()=>onMiss(item.objectID)}>按一下我删除</Button>  
 </div>  
 )}  
 </div>  
 )  
 }  
}  
//定义Button组建  
**class** Button **extends** Component  
{  
 render()  
 {  
 **const** {children,onClick}=**this**.props;  
 **return**(  
 <div>  
 <button  
 onClick={onClick}  
 className=''  
 >  
 {children}  
 </button>  
 </div>  
 )  
 }  
}  
**export default** App2;