上拉刷新

Scroll组件

if(this.**pullUp**){ //是否上拉刷新  
 // 监听是否滚动结束  
 this.scroll.on('scrollEnd', () => {  
 // 判断滚动的距离是否小于最大滚动距离加上50差值，负值  
 if (this.scroll.**y** <= (this.scroll.**maxScrollY** + 50)) {  
 // 滚动到底部就派发事件  
 this.$emit('scrollToEnd')  
 }  
 });  
}

**watch**:{  
 // 监听data数据，数据名与函数名要相同,data发生变化时重新计算高度,延迟进行是因为要等DOM渲染完毕  
 data(){  
 setTimeout(()=>{  
 this.refresh();  
 },20)  
 }  
}

**watch**:{  
 // 监听data数据，数据名与函数名要相同,data发生变化时重新计算高度,延迟进行是因为要等DOM渲染完毕  
 data(){  
 setTimeout(()=>{  
 this.refresh();  
 },20)  
 }  
}

Suggest组件

// 上拉刷新查找  
searchMore(){  
 if(!this.**hasMore**){  
 return;  
 }  
 this.**page**++;  
 search(this.**query**,this.**page**,this.**showSinger**,perpage).then(res=>{  
 if(ERR\_OK==res.**code**){  
 // 把请求回来的数据拼接到原来的数据中  
 this.**result**=this.**result**.concat(this.\_genResult(res.**data**));  
 this.\_checkMore(res.**data**);  
 }  
 })  
},  
// 检查是否还有更多数据  
\_checkMore(data){  
 const song=data.song;  
 if(!song.**list**.**length** || (song.curpage \* perpage) >= song.totalnum){  
 this.**hasMore**=false;  
 }  
},

Suggest组件优化

由于移动端在文本框输入内容时会使键盘弹起来，这样会使搜索会来的结果被键盘遮挡，所以这里我们要让better-scroll在滚动开始前让输入框失去焦点，这样就会使键盘收起来，不影响用户体验

if(this.**beforeScroll**){  
 // 监听滚动开始前事件  
 this.scroll.on('beforeScrollStart',()=>{  
 this.$emit('beforeScroll')  
 })  
}

blur(){  
 // 这里是让输入框失去焦点,目的是因为移动端在输入框中输入内容是，键盘会弹起来，输入框失去焦点时会使键盘收起来  
 this.$refs.**input**.blur();  
}

保存搜索历史到localstorage

import **storage** from 'good-storage';  
  
// 保存在localstorage中的key值  
const SEARCH\_KEY = '\_search\_';  
// 保存的最大数量，即可以保存多少条搜索历史  
const SEARCH\_MAX\_LENGTH = 15;  
  
// 往数组插入一个值，arr是数组，val是要插入的值，compare是比较器，即用来找到2个值是否相等，maxLength是数组最大长度。这个函数作用是把要插入的值放在第一位，并且去掉重复的  
function insertArry(arr, val, compare, maxLength) {  
 // 查找要插入的值是否在数组中  
 var index = arr.findIndex(compare);  
 // 插入的值已经在数组中，并别在第一个位置  
 if (index == 0) {  
 return  
 }  
 // 要插入的值在数组中，但是不在第一个位置  
 if (index > 0) {  
 // 删除已经存在的值  
 arr.splice(index, 1);  
 }  
 // 插入要插入的值  
 arr.unshift(val);  
 if (maxLength && arr.**length** > maxLength) {  
 // 删除最后一个元素  
 arr.pop();  
 }  
}  
  
// 保存搜索历史记录，并保存到localstorage中，query是搜索记录  
export const saveSearch = function (query) {  
 // 获取localstorage中key为\_search\_的值，有就直接返回，没有就直接返回默认的[]  
 let searches = storage.get(SEARCH\_KEY, []);  
 // 插入搜索记录  
 insertArry(searches, query, (item) => {  
 return item == query  
 }, SEARCH\_MAX\_LENGTH);  
 // 修改localstorage中的值  
 storage.set(SEARCH\_KEY, searches);  
 return searches;  
}