1. 数据处理

因为获取回来的数据不是我们想要的，所以要进行处理

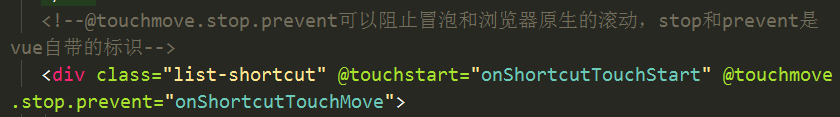
// 对请求的数据进行处理  
\_normalizeSinger(list){  
 var map={  
 **hot**:{  
 **title**:HOT\_NAME,  
 **item**:[]  
 }  
 }  
 list.forEach((item,index)=>{  
 // 获取前10条数据作为热门数据  
 if(index<HOT\_NAME\_LENGTH){  
 map.**hot**.**item**.push(new Singer(item.Fsinger\_mid,item.Fsinger\_name))  
 }  
 // 获取字母  
 var key=item.Findex;  
 if(!map[key]){  
 // 没有这个key就创建一个  
 map[key]={  
 **title**:key,  
 **item**:[]  
 }  
 }  
 map[key].**item**.push(new Singer(item.Fsinger\_mid,item.Fsinger\_name))  
 });  
 // 存储热门数据  
 var hot=[];  
 // 存储a-z数据  
 var ret=[];  
 for (var key in map){  
 // 获取map中的每一个数据  
 var val=map[key]  
 if(val.**title**==HOT\_NAME){ //标题是热门的  
 hot.push(val);  
 }else if(val.**title**.match(/[a-zA-Z]/)){ //标题是字母的  
 ret.push(val);  
 }  
 }  
 // 进行排序，a，b分别是相邻的2个数据  
 ret.sort((a,b)=>{  
 // charCodeAt(0)返回字符串第一个字符的 Unicode 编码:  
 return a.**title**.charCodeAt(0)-b.**title**.charCodeAt(0)  
 })  
 // 将2个数组拼接成一个新的数组并返回  
 return hot.concat(ret);  
}

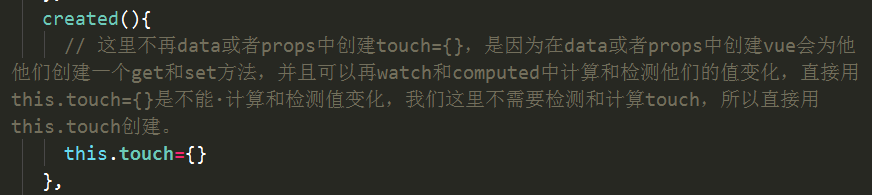
Singer类



2.歌手列表组件（listview）

A、右侧快速入口





onShortcutTouchStart(e){  
 // 得到我们点击的是元素的下标索引  
 var index=getData(e.**target**,'index');  
 // 获取第一个按下去的手指  
 var fistTouch=e.touches[0];  
 // 获取按下去的y值  
 this.**touch**.**y1**=fistTouch.pageY;  
 // 记录按下去的是第几个锚点  
 this.**touch**.**index**=index;  
 // 滚动到指定元素  
 this.\_scrollTo(index)  
},  
// 在右侧导航栏滑动时，是歌手列表也能滑动  
onShortcutTouchMove(e){  
 var fistTouch=e.touches[0];  
 this.**touch**.**y2**=fistTouch.pageY;  
 // 移动后相差几个锚点数量  
 var delta=(this.**touch**.y2-this.**touch**.y1) / HEIGHT | 0;  
 // 得到当前移动到第几个锚点  
 var index=parseInt(this.**touch**.**index)**+delta;  
 this.\_scrollTo(index);  
},  
\_scrollTo(index){  
 // 滚动到指定的元素  
 this.$refs.scrollContent.scrollToElement(this.$refs.scrollItem[index],0)  
}

在scroll组件中注册滚动监听事件

if(this.**listenScroll**){  
 var that=this;  
 // 监听滚动事件  
 this.**scroll**.on('scroll', (pros) => {  
 // 派发一个名为scroll的事件，即向父组件派发事件，向父组件通信  
 that.$emit('scroll', pros);  
 });  
}

父组件listview（滚动时，右侧变化）

<!--  
probeType派发滚动事件，不节流  
listenScroll注册滚动监听事件  
@scroll="scroll"接收scroll组件派发出来的scroll事件，处理函数是scroll，即子组件向父组件通信  
-->  
<scroll :data="**data**"  
 class="listview"  
 ref="scrollContent"  
 :probeType='**probeType**'  
 @scroll="scroll"  
 :listenScroll="**listenScroll**"  
>

// 处理scroll子组件派发过来的事件  
scroll(p){  
 this.**scrollY**=p.**y**;  
},  
// 计算高度  
\_calculateHeight(){  
 var height=0;  
 // 一开始高度为0  
 this.**listenHeight**.push(height);  
 for (var i=0;i<this.$refs.scrollItem.**length**;i++){  
 // 获取DOM的高度  
 var h=this.$refs.scrollItem[i].clientHeight;  
 height+=h;  
 this.**listenHeight**.push(height);  
 }  
}

// scrollY发生变化时，即发生滚动  
scrollY(newY){  
 // 当滚动到顶部时，newY大于0  
 if(newY>0){  
 this.**currentIndex**=0;  
 return;  
 }  
 // 在中间滚动时，this.listenHeight.length-1是因为listHeight比右边快速列表多一。  
 for (var i=0;i<this.**listenHeight**.length-1;i++){  
 // 上限，分组的顶端,相当于上一个分组的下限  
 var h1=this.**listenHeight**[i];  
 // 下限，分组的下端，相当于下一个分组的上限  
 var h2=this.**listenHeight**[i+1];  
 // !h2说明不是最后一个分组，即不是z，-newY因为滚动时为负值，这里要用正数比较  
 if(-newY>=h1 && -newY<h2){  
 this.**currentIndex**=i;  
 // 相当于减newY，因为newY是负值  
 this.**diff**=h2+newY;  
 return;  
 }  
 // 当滚动到底部时，切-newY大于最后一个元素上限  
 this.**currentIndex**=this.**listenHeight**.length-2;  
 }  
},

点击右侧导航栏是，锚点要发生变化

\_scrollTo(index){  
 // 由于点击事件是注册在父容器上，所以当我们点击的地方不是A-Z时，index是null  
 if(index==null){  
 return;  
 }  
 // 滚动时，由于是通过滚动的距离除每个DOM的高度算出index，但是超出右侧导航时会令index小于0或者大于导航的个数  
 if(index<0){  
 index=0  
 }else if(index>this.**listenHeight**.length-2){  
 index=this.**listenHeight**.length-2  
 }  
 // 由于点击时没有触发scrollY的变化，所以我们要手动设置  
 this.**scrollY**=-this.**listenHeight**[index];  
 // 滚动到指定的元素  
 this.$refs.scrollContent.scrollToElement(this.$refs.scrollItem[index],0)  
},

1. 固定标题

计算属性中

// 固定标题  
showTitle(){  
 // 当在顶端是返回空字符  
 if(this.**scrollY**>0){  
 return '';  
 }  
 return this.**data**[this.**currentIndex**]?this.**data**[this.**currentIndex**].**title**:'';  
}

监测属性

diff(newval){  
 // 标题要往上滚动的距离  
 var fixedTop=(newval >0 && newval <TITLE\_HEIGHT)?newval-TITLE\_HEIGHT:0;  
 // 这里不能用fixedTop==0判断，因为为0后返回，translate3d(0,${fixedTop}px,0)中的fixedTop一直都是0之前的值，不能归0返回原位，这里只有fixedTop和this.fixedTop都为0时才能相等，这里是为了节流  
 if(fixedTop==this.**fixedTop**){  
 return  
 }  
 this.**fixedTop**=fixedTop;  
 this.$refs.fixed.**style**.**transform**=`translate3d(0,${fixedTop}px,0)`;  
}