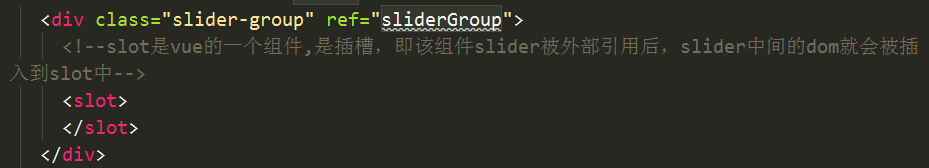
1. 轮播图

a、slot组件

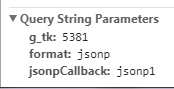
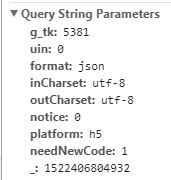


b、Jsonp封装(跨域)

// jsonp解决跨域问题  
import originJSONP from 'jsonp';  
//url是请求路径，data是参数，option是jsonp一些配置参数  
export default function jsonp(url,data,option) {  
 url+=(url.indexOf('?')<0?'?':'&')+param(data)  
 //Promise是es6的api，Promise成功后调用resolve，失败调用reject  
 return new Promise((resolve,reject)=>{  
 originJSONP(url,option,(err,data)=>{  
 if(!err){ //成功  
 resolve(data);  
 }else { //失败  
 reject(err);  
 }  
 })  
 })  
}  
//处理参数  
function param(data) {  
 var url='';  
 for (var key in data){  
 let value=data[key]!==undefined?data[key]:'';  
 //encodeURIComponent，把字符串作为 URI 组件进行编码，vlaue其中的某些字符将被十六进制的转义序列进行替换。  
 url+=`&${key}=${encodeURIComponent(value)}`  
 }  
 //url不为空时去掉第一个&  
 return url?url.substring(1):'';  
}

C、请求数据(qq音乐后台)

请求参数（通过开发者工具获取所需要的请求参数）



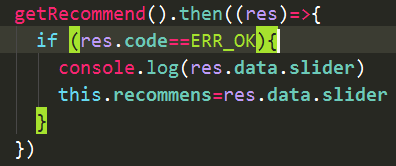
配置通用参数

// 请求的通用参数  
export const commonParams={  
 **g\_tk**: 1928093487,  
 **inCharset**: 'utf-8',  
 **outCharset**: 'utf-8',  
 **notice**: 0,  
 **format**: 'jsonp'  
}  
// jsonp的配置参数  
export const options={  
 **param**:'jsonpCallback'  
}  
// 请求是否成功，0是成功  
export const ERR\_OK=0

发送请求

export function getRecommend() {  
 // qq音乐  
 var url='https://c.y.qq.com/musichall/fcgi-bin/fcg\_yqqhomepagerecommend.fcg';  
  
 // 拼接成一个对象  
 const data=Object.assign({},commonParams,{  
 **g\_tk**:5381,  
 **platform**: 'h5',  
 **uin**: 0,  
 **format**:'json',  
 **needNewCode**: 1,  
 **\_**:1522572975899  
 });  
 // 返回一个Promise对象  
 return jsonp(url,data,options)  
}

处理数据



1. 歌单列表

请求数据

不同于轮播图，因为请求歌单数据被限制了，所以无法通过前端发送请求获得数据，我们可以通过本地服务器（webpack.dev.conf）获取去数据，然后前端访问本地服务器，本地服务器吧请求回来的数据响应给前端

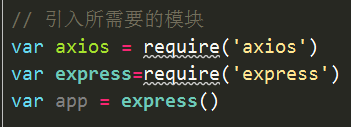
限制请求数据原因



要求访问qq音乐后台的必须是指定的网页

所以我们通过本地服务器模拟欺骗qq音乐后台使他误以为发出请求的是指定网页

第一步



第二步，找到devServer

const devWebpackConfig = merge(**baseWebpackConfig**, {  
 **devServer**: {  
 before(app) {  
 // 前端访问本地服务器  
 app.get('/getDiscList', function (req, res) {  
 var url = 'https://c.y.qq.com/splcloud/fcgi-bin/fcg\_get\_diss\_by\_tag.fcg'  
 // 发送请求  
 axios.get(url, {  
 // 请求头，欺骗qq音乐后台，使它误认为发出请求的是https://c.y.qq.com/  
 **headers**: {  
 **referer**: 'https://c.y.qq.com/',  
 **host**: 'c.y.qq.com'  
 },  
 // 请求所需要的参数,axios会自动拼接成xx=xx&yy=yy  
 **params**: req.**query** }).then((response) => {  
 // 请求成功后返回数据给前端  
 res.json(response.**data**)  
 }).catch((e) => {  
 console.log(e)  
 })  
 })  
 },  
 })

前端访问本地服务器

export function getDiscList() {  
 const url = '/getDiscList'  
 const data = Object.assign({}, commonParams, {  
 **platform**: 'yqq',  
 **hostUin**: 0,  
 **sin**: 0,  
 **ein**: 29,  
 **sortId**: 5,  
 **needNewCode**: 0,  
 **categoryId**: 10000000,  
 **rnd**: Math.random(),  
 **format**: 'json'  
 })  
 return axios.get(url, {  
 **params**: data  
 }).then((res) => {  
 return Promise.resolve(res.data)  
 })  
}

显示歌单列表数据

注意事项:

1.better-scroll只对容器下的第一个子节点进行滚动。

2.Better-scroll横向滚动时要有宽度，否则不会滚动，纵向滚动时也要有高度

3.因为我们发出了2次数据请求，所以在轮播图有一张图片加载完成后就强制better-scroll重新计算高度，同时把歌单列表数据传递进我们自定义的scroll组件中，并在scroll组件内监听该属性是否变化并让better-scroll重新计算高度。这样推荐页面整体的高度才能正确计算出来，才能进行滚动。即只要有数据变化就需要让better-scroll重新计算

4.因为我们多次用到滚动，所以我们抽象出一个滚动组件。

5.DOM要生成了才能初始化better-scroll，否则会报错。

6.由于fastclick和better-scroll（click属性）会发生冲突，导致点击后不能跳转，所以我们需要在img的class中添加needsclick，这样当我们点击图片时，点击事件就不会被拦截。

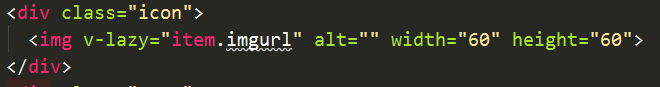
7.要滚动的内容高度或者宽度必须要比容器的高度或者宽度大，否者无法滚动。

8.better-scroll在y轴滚动的距离，不动时滚动距离为0，在顶端往下拉时滚动距离为正数，其他情况都为负数

图片懒加载（vue-lazyload）

用法：





官网例子





