一、

1.api文件夹：存放各页面组件利用jsonp跨域获取接口的jsonp函数包装

* config.js：定义了jsonp所需要的参数对象commonParams，主要包括一些公共属性g\_tk，format，inCharset等，还定义了一个options对象参数，包含param属性，最后定义了返回成功的常量ERR\_OK = 0
* recommend.js：（jsonp方式获取）getRecommend函数，用于获取轮播图的所有信息，（node的express获取）getDiscList函数，用于获取热门歌单推荐列表信息
* sing.js：getSingerList函数，用于获取歌手列表，getSingerDetail函数，用于获取歌手详情页信息
* rank.js：用于请求rank组件需要的数据，定义了getTopList函数，请求排行榜的数据，定义getMusicList函数，获取排行榜的详情页信息
* search.js：getHotKey函数，用于获取热门推荐的列表，search函数，用于获取搜索结果的列表信息
* song.js：getLyric函数，获取歌词信息

2.base文件夹：存放各个基础组件

comfirm组件：

* 设置showFalg的boolean值（初始值为false）来控制弹框是否显示
* 当调用show函数时，将showFalg设为true，调用hide函数时，设为false
* 在清空按钮上绑定confirm事件按，向外派发确认事件，在取消按钮上绑定cancel事件，向外派发取消事件

list-view组件：

* 调用scroll组件，使歌手列表滚动，设置listenScroll值为true，绑定scroll事件，试试更新scrollY的值，将probeType属性设置为3
* 在歌手的li上绑定selectItem点击事件，用于向外派发selet事件，点击进入详情页
* 依赖懒加载，v-lazy，使图片懒加载
* 定义\_calculateHetght函数，计算滚动的高度，设置一个空数组listHeight和一个初始值为0的height常量，获取整个歌手列表，遍历每个歌手li，取得li的高度，并复制给height常量，在存到数组里，这样每个数字想就是li所在高度，便可知道是哪个li
* 定义滚动函数\_scrollTo函数，如果index参数<0，则将index设置为0，如果index>立标的li长度，就把它设置长度-2，函数中要设置scrollY的值等于-this.listHeight[index]（因为向上滚，所以是负值），最后调用scroll组件的scrollToElement事件，使列表滚动到相应位置
* 给右侧的字母索引块添加触摸事件

onShortcutTouchStart函数：传入当前触摸点，调用common/js/dom中的getData函数，设置当前触摸点的索引值，获取当前触摸点的纵坐标，将纵坐标的索引值都加入定义的空对象touch中，最后调用\_scrollTo函数，使列表在当前触摸点的位置

onShortcutTouchMove函数：再次获取此时触摸点的纵坐标，用此时的纵坐标减去刚才的纵坐标，所得差值在除以没个li的高度，便可知道滚动了几个li，再将刚才的索引值加上滚动了的li的个数，便得到此时的索引值，最后调用\_scrollTo函数

* 给每个字母索引绑定class，当currentIndex===index条件满足时，使字母高亮
* 定义shortcutList函数，利用map函数和substr方法生成每块歌手上面的字母标题
* 定义fixedTitle函数，设置顶部显示的字母标题
* 在watch里监听diff方法，如果参数newVal在0—顶部标题高度30之间，则用newVal-30，以此获得标题快需要向上滑动的距离，最后利用thansform做一个向上滑动的3D效果
* 在watch中监听scrollY方法，如果参数newY大于0，说明是在向下拉，此时设置索引值为0，利用上面得出的高度数组，判断，如果newY在上一个li和下一个li之间，则将索引值设为上一个的索引值，并将diff的值设为height2 + newY（因为newY是负值，所以是加号）

loading组件：

* 这是一个，在网速较慢，加载的数据为渲染是调用的组件，所以显示条件是数据的length为0时

no-result组件：

* 这是一个返回的数据为空时，显示的组件

progress-bar组件：

* 这是进度条的组件
* 设置props的percent属性，表示播放时间和总时间的百分比
* 定义\_offset函数，传入参数offsetWidth，用来设置已播放时间的div宽度和可拖动的圆点的thansform动画
* 定义\_triggerPercent函数，设置persent，并向外派发percentChange事件
* 定义progressTouchStart函数，获取触摸点的横坐标和已播放时间的可见宽
* 定义progressTouchMove函数，获取此时触摸点的横坐标，求得两次横坐标的差值deltaX（为负），用总的时间长减去差值，就是剩下的时间长，调用\_offset函数，将剩的时间长作为参数传入函数
* 定义progressTouchEnd函数，直接调用\_triggerPercent函数，计算百分比
* 定义progressClick函数，实现点击进度条调整播放进度的功能，利用getBoundingClientRect()方法获取整个进度条相对于浏览器左边的距离，获取点击点的横坐标，用横坐标减去刚才的距离便是已播放的时间长，最后调用\_offset函数计算宽度和\_triggerPercent函数计算百分比
* 在watch中监听percent方法，接受参数newPercent，用newPercent与总的进度条相乘就是播放过的时间，在调用\_offset函数

progress-circle组件：

* 这是mini播放器的圆形进度条组件

scroll组件：

* 引入BScroll，对BScroll做一些初始化处理
* props定义了：

1.probeType: 1 滚动的时候会派发scroll事件，会截流。2滚动的时候实时派发scroll事件， 不会截流。 3除了实时派发scroll事件，在swipe的情况下仍然能实时派发scroll事件

2.click: true 是否派发click事件，通常判断浏览器派发的click还是betterscroll派发的click，可以用event.\_constructed，若是bs派发的则为true

3.data：传入滚动组件的数据

4.listenScroll：是否监听scroll事件

5.pullup：是否允许上拉

6.beforeScroll：是否向外派发beforeScroll事件

* 定义\_initScroll函数，初始化scroll，new 一个BScroll对象，在允许监听滚动的条件下，监听scroll，并向外派发scroll事件，在允许上拉的时候，监听scrollEnd，并向外批发scrollEnd事件，在允许监听滚动之前的事件的条件下，派发beforeScroll事件
* 初始化Bscroll的基础事件，即调用它的enable，disable，refresh，scrollTo，scrollToElement
* 在watch里面监听更新事件，设置定时器，进行实时更新

search-box组件：

* 这是搜索页的搜索框组件
* 利用v-model进行输入框input的双向绑定
* 给清除标志添加clear事件，将搜索框的内容清空
* 定义setQuery函数，设置搜索框内容
* 定义blur函数，调用js的blur函数，使input失去焦点
* 监听搜索框的内容变化，引入common/js/util的debounce，定时向外派发query更新事件

search-list组件：

* 将ul标签改为<transition-group name="list" tag="ul">，以保证在删除列表项时有平稳的滑动，此时li要绑定kry值，即：key=”“
* 在li上绑定selectItem函数，以此向外派发select事件
* 在删除符号的icon上绑定deleteOne事件，以此向外派发delete事件
* 设置props的searches属性，有父组件传入

slider组件：

* slider组件是实现轮播的组件
* 利用slot插槽，有父组件传入dom结构
* 利用：class绑定，设置轮播点dots的active状态
* 它的props接收两个属性loop：是否循环播放 autoplay：是否自动播放
* 由于轮播的 容器会根据浏览器大小设定，所以需要动态设置，在nethods里定义一个设置宽的函数，主要功能是：（1）获取slider容器的可见宽，获取所有滚动li，遍历li的宽度为可见宽

（2）将每个li的宽度累加再\*2所得值复制给slider容器的ul的宽

* 在methods定义初始化slider的函数：

（1）importbetter-scroll，并定义一个BScroll的实例对象this.slider,其中snap参数是slider专用参数

（2）监听滚动结束scrollEnd事件：利用BScroll getCurrentPage事件获取当前滚动页，在循环模式下，将当前滚动页-1，将所得值设置为当前滚动页得索引，在自动播放模式下，执行播放函数

* 在methods定义自动播放的函数：

（1）获取初始化slider函数中的当前页索引值，并+1

（2）在循环循环模式下，索引要不断+1（用if条件判断是否loop）

（3）设置定时器，利用BScroll的goToPage函数，让li根据索引值轮播

* 在methods定义初始化dots圆点的函数：

（1）定义一个数组对象的实例this.dots，长度为li的个数

song-list组件：

* 这是歌手详情页，排行榜详情页都需要引入的组件
* 给list的li绑定selectItem事件，用于向外派发select事件
* 判断是否在排行榜详情，决定是否显示排行标志的div，给div的span绑定一个class，如果是前三名，则加上前三名的样式的类名，否则加上其他名次的样式类名，给div的span的内容调用一个getRankText的函数，如果歌曲的 index大于二，就显示index+1的数字
* 定义getDesc，用于描述span的内容，是内容是歌手+专辑的格式

switches组件：

* 这是tab切换效果的组件，用于像添加歌曲页面的最近播放和搜索历史这样的tab切换
* 在li上绑定switcheItem事件，向外派发switche事件
* 在props定义currentIndex属性，用于绑定active状态的class，当currentIndex === index即active状态

top-tip组件：

* 这是顶部提示的组件，用在添加歌曲页面
* 定义showFlag，为true时组件显示，为false组件隐藏
* 定义show方法，设置计时器，使组件在规定时间内显示之后，调用hide方法，自动消失
* 定义hide方法，绑定在div上，使得点击div后组件隐藏

3.common/js文件

cache.js：

* 调用good-storage本地存储
* 定义insertArray方法，实现歌曲的插入，利用splice（）让每个歌曲项向前一项，再利用unshift（）把新的歌曲插入歌曲数组

config.js：

* 定义一个播放模式的对象

dom.js:

jsonp.js：

* 引入依赖的jsonp
* 定义一个jsonp函数，在函数中new一个promise实例，在实例中调用jsonp，如果返回成功，就resolve，不成功就reject
* 最后返回promise

mixin.js：

* 存放复用性强，几个组件都会用到的代码
* 包括mini播放器显示时，列表bottom样式的动态变化函数
* 包括播放器收藏，切换播放模式的函数
* 包括搜索页面搜索框和搜索历史的函数

singer.js：

* 定义一个singer的类，包括id，name，avator

song.js：

* 定义一个song类，包括获取歌词，以及返回歌词的形式
* 定义一个creatSong的函数，调用song类，将送类需要的参数传入

util.js：

* 定义shuffle函数，用于歌曲列表打乱歌曲顺序，在随机播放模式下调用

二、

1.recommend页面组件：

（1）引入recommend.js，用于获取轮播图数据和下面的列表数据

（1）getRecommend：获取轮播数据，将返回值的slider复制给data的recommends

（2）getDiscList：获取推荐列表，将返回值的list复制给data的discList

（2）引入slider.vue组件，实现轮播图和下面的原点映射

（3）在li上添加点击事件，进入详情页：

* 将当前li项作为参数传给点击函数，利用vue的this.$router.push（），将路由指向详情页路径/recommend/item.id
* 引入vuex的 mapMutations，在store中有SET\_DISC的mutation函数，用于渲染详情页的头部标题和背景图

2.singer歌手页面：

（1）引入基础组建list-view：

* 在计算属性中返回旁边的字母索引和每一块的字母标题
* 在props设置需要外部传入的data
* 由于list-view是基础组件，所以没有逻辑处理的代码，所以点击歌手li时，有一个seletcItem点击事件，只是向外派发一个select事件
* 在methods中定义onShortcutTouchStart函数，用于获取开始触点的位置，及所在索引，将索引传递给\_scrollTo函数
* 在methods中定义onShortcutTouchMovet函数，用于获取触点移到的位置和所在的字母块，及所在索引，将索引传递给\_scrollTo函数
* 在methods中定义 \_calculateHetght函数，用于设置歌手li的宽度
* 在methods中定义\_scrollTo函数，用于是列表滚动指定的索引位置
* 在watch里面监听歌手li的高度，设置计时器，实时更新高度，以防浏览器改变印象布局
* 在watch里监听scrollY，当滚动距离在一个li中间高度的时候，将歌手索引设置为前一个歌手的索引，并设置diff（表示当前歌手li出去滚动的高度外，还是多少高度）
* 在watch里面监听diff的变化，并利用tansform设置字母表题在凉快快要重合时的滑动效果

（2）在methods中定义list-view抛出的点击事件select的函数selectSinger：利用vue的this.$router.push，将路由指向歌手详情的链接，并利用引入vuex的mapMutations的SET\_SINGER方法渲染详情页面

（3）利用api文件夹中singer.js封装的jsonp获取歌手列表，将数据返回给data，并传给list-view组件

（4）在methods中定义\_normalizeSinger函数用于处理获取到的打他数据：数据处理分为热门数据和按字母分类的数据，将热门歌手放进hot数组，一般歌手放进ret数组，在利用sort（）将ret进行排序

3.rank排行页面

（1）引入api的rank.js封装的jsonp获取数据，并返回给data函数

（2）引入Scroll基础组件，将返回的数据传入，使排行列表滑动

（3）在li上绑定selectItem点击事件，跳转到详情页，此时同样要利用this.$router.push和maoMutations，实现跳转和渲染

4.search搜索页面：

（1）页面的搜索栏是引入searchBox组件：

searchBox组件：

* 定义一个搜索框，给定placeholder默认值
* v-model双向绑定搜索框的内容
* 在清除符号上绑定清空搜索框内容的点击事件
* 在methods中定义blur方法，是搜索框失去焦点
* 引入common/js/util中的debounce，监听搜索框的内容变化，并实时派发内容改变的函数

（2）引入api的search.js封装的jsonp，获取热门推荐的数据

（3）引入suggest组件，显示搜索结果列表：

suggest组件：

* 引入Scroll基础组件，并传入data，pullup，beforeScroll等属性值
* 引入common/js/song.js的createSong函数，并在methods中定义一个\_normalizeSongs函数将传入的歌曲数据实例化为一个song类
* 在methods中定义一个\_genResult函数，运用vue的扩展符和\_normalizeSongs方法将歌曲数据规范为歌手：xxx；专辑：xxx的类似形式
* 在methods中定义一个search函数，引入api/search.js封装好的jsonp获取数据，并调用\_genResult函数对数据进行规范，得到想要的形式
* 由于suggest是基础组件，所以在搜索列表的li上绑定click事件selectItem，向外派发select时间，如果搜索出歌手，就路由到歌手详情页，并引入vuex的mapMutations中的SET\_SINGER渲染页面，如果搜索出歌曲，则引入vuex的mapActions的insertSong，渲染歌曲详情页面
* 在搜索结果列表中得i标签绑定class的判断函数getIconCls，判断歌手还是歌曲，来选择渲染的图标
* 在搜索结果的name上绑定getDisplayName函数，并用v-html转义，如果是歌手，就返回歌手名，如果是歌曲就返回歌曲名+歌手名
* 在methods里定义\_checkMore函数，用于检查是否还有更多数据，在song.curnum + song.curpage \* perpage（perpage是每页长度）大于song.totalnum（totalnum是总的数据数）时，将定义的hasMore设置为false
* 在scroll组件上绑定searchMore函数，用于加载下一个搜索结果，在函数中使page++，并调用search方法，获取下一页数据，并调用\_checkMore方法检查是否还有更多数据
* 在watch中实时监听搜索列表诗句query的变化，即调用search方法获取数据

（4）在推荐的li上绑定一个addQuery事件，该函数调用searchBox派发的setQuery事件，将点击的li内容更新到搜索框中

（5）搜索历史块，显示条件是searchHistory.length

（6）同样在搜索历史的li上添加addQuery事件，将内容更新到搜索框

（7）给搜索历史的清空垃圾桶的标志添加一个显示确认清空弹框的点击事件

（8）引入search-list基础组件来显示搜索历史列表

（9）定义search-list派发的select事件安慰addQuery，delete事件为deleteSearchHistory

（10）引入base/confirm/confirm组件，设置确认清空历史记录的弹框

（11）向comfirm组件传入confirm事件的方法clearSearchHistory

（12）在watch中监听query的变化，设置定时器实时更新

5.add-song组件：

* 引入switches组件，做最近播放和搜索历史的切换效果
* 引入scroll组件，使播放列表和历史列表轮播
* 引入suggest组件，显示搜索结果列表
* 引入top-tip组件，显示添加成功的提示

6.music-list组件：

* 这是点击歌手后进入的页面（逻辑页面，没有传入具体数据）
* 引入usong-list基础组件，渲染歌曲列表
* 引入scroll组件，使歌曲列表滚动
* 在watch里面实时监听更新scrollY的变化，还要在上拉时改变filter的显示，背景蹄片的高度，和song-list组件div的top值
* 引入高级css的backdrop-filter效果，展示上拉时背景的平滑效果

7.disc组件：

* 是排行榜歌曲页面的详情页
* 引入music-list页面，传入具体数据渲染页面的组件
* 引入recommend的api封装的jsonp，获取歌曲列表

8.play-list组件：

* 这是点击mini播放器的列表icon之后出现的页面
* 有comfirm确认是否清空
* 引入add-song组件，是add-song组件的入口

9.player组件：

* 这是播放器的页面，包括playlist页面和迷你播放器的缩放
* 空能有播放暂停切换，歌曲切换，歌词页面的切换

10.singer-detail组件：

* 是歌手组件的详情页，调用了music-list组件

11.tab组件：

* 四个大组件的tab切换组件
* 利用router-link实现tab切换

12.top-list组件：

* 排行榜点击之后进入详情页的组件
* 调用music-list

三、router文件夹

* 采用router的异步加载路由的方式避免损耗

const Recommend = (resolve) => {

import('components/recommend/recommend').then((module) => {

resolve(module)

})

}

* 自路由用children路由

{

path: '/rank',

component: Rank,

children: [

{

path: ':id',

component: TopList

}

]

}