# Vue学习第七天

## 反馈

## 回顾

## player-歌曲搜索

### 01 - 搜索路由

1. 在components里面声明组件`SongList.vue`
    1. 把results.html的htmlcopy到SongList.vue
2. 在main.js里面引入SongList.vue
3. 添加路由规则

```
{
  path: '/search',
  component: SongList
}
```

4. 改变hash看是否能访问到SongList.vue

### 02 - 搜索路由切换

输入关键字，回车/点击搜索按钮，显示搜索结果列表组件

1. 获取用户输入的关键字 v-model:keywords
2. 事件 @keyup.enter/@click:toSongList
3. js的方式去改变hash为/search。编程式导航

```
this.$router.push('/search')
```

## 注意点

1. 为什么子组件里面可以能过this.$router拿到路由的实例
    1. import from 'node_modules文件夹名'，实际上引入的是文件夹下package.json里面main属性指向的js
    2. 子组件方法中的this访问$router时，其实就是拿到路由实例
    3. Vue实例上访问属性，如果找不到的话，会去prototype上去找

```
 Object.defineProperty(Vue.prototype, '$router', {
   get: function get () { return this._routerRoot._router }
 });
```

```
<script>
  function Vue() {

  }
  Vue.prototype.$router = {
    push() {
      console.log('能访问到$router的push方法')
    }
  }

  const app = new Vue()
  // Vue实例上访问属性，如果找不到的话，会去prototype上去找
  app.$router.push()

  app.name = 'Joven' //对象上添加属性
</script>
```

## 03 - 搜索路由传参

动态路由匹配:组件从hash里获取参数

1. Search.vue参数传到hash里面

```
this.$router.push('/search/keywords')
```

2. main.js 修改路由规则

```
path: '/search' ==> '/search/:keywords'
```

3. 在Vue开发工具里面查看搜索结果列表组件的数据$route

### 注意点

1. this.$router是路由实例与this.$route这是路由数据

## 04 - 饿了么ui 为空消息提示

传送门

使用方法

```
this.$message('这是一条消息提示');
```

注意点：

1. 为什么Vue实例可以访问到$message

```
Vue.prototype.$loading = packages_loading.service;
Vue.prototype.$msgbox = message_box;
Vue.prototype.$alert = message_box.alert;
Vue.prototype.$confirm = message_box.confirm;
Vue.prototype.$prompt = message_box.prompt;
Vue.prototype.$notify = notification;
Vue.prototype.$message = packages_message;
```

## 05 - 渲染搜索结果

尽早发请求，获取数据，渲染搜索结果

1. created里面发请求
2. axios.get() https://autumnfish.cn/search?keywords=%E6%B5%B7%E9%98%94%E5%A4%A9%E7%A9%BA
3. 关键词在路由数据里面
4. songs结合v-for遍历数组，渲染列表
5. mv图标的展示 mvid不为0时

## 06 - 过滤器歌手处理

自定义过滤器formatSinger

1. 定义过滤器
2. 使用

```
{{数据|formatSinger}}
```

filters:formatSinger(singers){ return}

## 07 - 过滤器时间处理

自定义过滤器formatTime

1. 使用

```
{{数据|formatTime}}
```

2. 定义过滤器
    1. 组件的filters属性里面声明一个formatTime的方法
    2. formatTime方法接受一个time，time就是过滤器作用的数据
        3. 数据处理 毫秒->04:03
            1. 总秒数 = 毫秒/1000
            2. 分= Math.floor(总秒数/60) 6. 秒= Math.floor(总秒%60) 7. return数据处理结果，是最终渲染结果

# 侦听器

传送门

侦听器就是用来监听data属性的变化

1. 使用方法
    1. watch和el、data是平级的属性
    2. 在watch里面，要监听的属性是一个方法，方法名为属性名
    3. 如果要监听的属性是不符合变量规则，比如有-或者.的话，加引号
2. watch相比于updated，updated是所有的data属性改变都会触发，watch可以自定义一些属性改变时触发

```
watch: {
  // message: function () {

  // }
  message() {
    console.log('改变了')
  },
    'user.name'() {
      console.log('改变了')
    }
}
```

## 09 - 再次搜索bug修复

1. 当search改变时重新调用接口
2. created中已经实现了接口调用
3. 抽取为函数，在2个地方调用即可

### 注意

1. created组件如果不被销毁，只会触发一次
2. 如果有需求在特定数据改变时重新执行逻辑，可以使用 侦听器`watch`
3. watch和updated相比触发的频率 低

# player - 播放歌曲

## 01 - 点击去播放器

1. 在SongList.vue歌曲列表的 左侧播放按钮上绑定 点击事件
2. 点击事件中获取歌曲的id
3. 触发之后，使用编程式导航跳转去播放器 放歌，携带id
4. 动态路由匹配 main.js
    1. 创建04.player.vue组件
    2. path:'/player/:id'
    3. component:player

# 02-Vue全局使用axios与axios基地址设置

1. Vue全局使用axios
    1. 在main.js中把axios设置给Vue的原型

```
Vue.prototype.$axios = axios
```

    2. 所有组件可以访问到axios

```
this.$axios
```

2. 设置axios基地址，使用axios的地方url可以略基地址
    1. 在mian.js设置axios的基地址

```
axios.defaults.baseURL = '设置的基地址';
```

## 注意

1. $的目的是和自己的属性区分，这是一个大伙都遵守的约定
2. 设置了之后，所有的组件内部都可以通过`this.$axios`访问axios
3. axios设置了基地址之后，请求有2种情况
    1. 请求的地址是完整的,`https://autumnfish.cn/banner`
        1. 不会去拼接基地址
    2. 请求的地址只有一部分：`/banner`
        1. axios就会自动补全基地址部分

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <title>Document</title>
</head>

<body>
```

```
  <div id="app">
    <button @click="search">查询banner</button>
  </div>
  <script src="./lib/vue.js"></script>
  <script src="./lib/axios.js"></script>
  <script>
    // 把axios设置给Vue的原型
    Vue.prototype.$axios = axios
    // 设置基地址
    axios.defaults.baseURL = 'https://autumnfish.cn';

    const app = new Vue({
      el: "#app",
      data: {},
      methods: {
        search() {
          // 如果url是相对地址，url=基地址+相对地址
          // 如果url是完整的，  不会拼接基地址
          this.$axios.get('https://autumnfish.cn/banner')
            .then(res => {
              console.log(res)
            })
        }
      },
    });
  </script>
</body>
</html>
```

## 03-Vue全局使用axios与axios基地址设置整合

## 03 - 歌曲信息显示

## 04 - 歌曲url获取

## 05 - 歌词显示

处理歌词用的正则`/\[\d{2}\:\d{2}\.\d{2,3}\]/,`

上面的 3 4 5步骤类似

created中 调用 歌曲url接口，歌曲封面接口，及 歌词接口即可

# player - 歌曲评论

## 01 - 歌曲评论路由设置

1. 在components里面声明组件`Comment.vue`
    1. 把comment.html的html copy到`Comment.vue`
2. 在main.js里面引入Comment.vue
3. 添加路由规则

```
{
  path: '/comment',
  component: Comment
}
```

## 02-点击携带歌曲id去评论组件

1. 在 SongList.vue组件中，添加点击事件
    1. @click:toComment
2. 为歌名 绑定点击或者双击事件(dblclick)
3. 编程式导航`this.$router.push('/comment/${id}')`

### 注意点

1. 根据需求找到设置的文件即可

## 03 - 获取评论信息

1. created获取评论信息 /comment/hot?id=186016&type=0
2. then方法中
3. data中加数据

4. 页面中写vue指令 渲染

## 04 - 格式化评论时间

1. 下载moment.js
    1. npm i moment
2. 05.comment.vue中添加一个过滤器
    1. filters:{ formatTime(time){ 处理并返回 } }
3. {{ 数据 |过滤器}}
4. 时间的处理，第一时间想到moment这个库

# player - mv播放

## 01 - mv路由设置

1. main.js
    1. 新建06.mv.vue
    2. path:"/mv/:mvid"
    3. component:mv

## 02-点击携带mvid去mv组件

1. 04.results.vue中 为mv的图标绑定点击事件 传入id
    1. this.$router.push('/mv/${id}')

## 03 - 获取mv信息

1. created中调用接口 获取数据 /mv/detail?mvid=5436712

## 04 - 播放最高清的mv

1. then中获取清晰度最高的mv进行播放

### 注意点

1. 对象的属性获取可以用点语法也可以中括号的语法

# 总结