Vue学习第4天

反馈

回顾

Demo-播放器

接口文档地址

实现步骤

1. 输入关键字，回车，搜歌，展示歌曲列表
    1. 获取关键字 v-model.keyword
    2. 回车 @keyup.enter:queryWeather
    3. 搜歌 axios.get(`<https://autumnfish.cn/search?keywords=海阔天空>`)
    4. 渲染列表vfor:songList
2. 双击一首歌，播放歌曲
    1. 接口 https://autumnfish.cn/song/url?id=444267215
    2. 双击一首歌 @dblclick：playMusic(item.id)
    3. 播放歌曲audio :src="musicUrl"
3. 双击一首歌更新封面
    1. playMusic方法里面获取封面
    2. 接口 https://autumnfish.cn/song/detail?ids=347230
    3. 显示图片v-bind:src="picUrl"
4. 双击一首歌更新热评
    1. playMusic方法里面获取热评
    2. 接口 https://autumnfish.cn/comment/hot?id=186016&type=0
    3. 渲染列表vfor:commentList

注意点

1. 实际工作中，前端除了做界面就是调不同的后端接口。
2. 网易云音乐的接口数据稍复杂，没关系，只要一层一层地拨，肯定可以找到我们想要的数据。

Demo-播放器

实现步骤

1. 有mv的歌曲就展示mv超链接，点mv超链接在新窗口打开mv
    1. 在搜索结果里，判断每一首歌mvid是否为0，不为0的话，展示mv超链接
    2. 点击mv链接 @click:playMV
    3. 获取mv的地址
        1. axios.get
        2. 接口 https://autumnfish.cn/mv/url?id=5436712
    4. 在新窗口中打开mvUrl window.open(mvUrl)

2. 点击播放，杆放下，碟片和封面旋转; 点击暂停，杆抬起，碟片和封面停止旋转
    1. 杆放下，碟片和封面旋转，所在的元素上添加playing类名
    2. 杆抬起，碟片和封面停止旋转，所在的元素上移除playing类名
    3. 设置一个是否在播放的状态的变量isPlaying，能过v-bind来控制playing类名 v-bind:class="{playing:isPlaying}"
    4. audio 暂停时 onpause事件 isPlaying=false
    5. audio播放时 onplay事件　isPlaying=true

3. 列表动画
    1. 包裹列表，Vue才会添加动画 transition-group name="类名的首单词" tag="ul"
    2. li v-for 需要给key
    3. 在Vue添加的类名里，自定义样式。用官方例子的样式，copy粘贴然后修改
    4. 让列表元素依次显示，设置transition-delay
    5. 为了让每次搜索结果都有动画，需要清空搜索结果数组
    6. 不需要元素离开的动画

注意点

1. a标签是有默认事件的，所以点击事件应该加上@click.prevent阻止默认事件
2. audio里面的pause事件和play事件，可以通知我们当前是播放状态还是暂停状态。
3. 动画钩子函数leave,是元素消息失的动画过程中的钩子函数，leave回调里面，直接调用done()，就让动画结束

iScroll

传送门

中文文档传送门

iScroll是一个优秀的javascript滚动插件。

使用方法

1. 导包，引入iscroll.js
2. 三层嵌套的结构,最外层的标签的高度应该要比里面内容要小

```
<div id="wrapper">
      <ul>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
```

3. 实例化IScroll

```
//构造函数传选择器
var myScroll = new IScroll('#wrapper');

//构造函数传dom
var wrapper = document.getElementById('wrapper');
var myScroll = new IScroll(wrapper);
```

注意点

refresh方法告诉iscroll内容的高度发生了变化。

否则新添加的元素，无法滚动。

```
myScroll.refresh();
```

```
<!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>iscroll的基本使用</title>
    <style>
      #wrapper {
        width: 400px;
        height: 200px;
        border: 1px solid green;
        /* overflow: hidden; */
      }
    </style>
  </head>
  <body>
    <button onclick="add()">添加</button>
    <div id="wrapper">
      <ul>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
        <li>是兄弟，就是一起来玩贪玩蓝月</li>
      </ul>
    </div>
```

```
    <script src="./lib/iscroll.js"></script>
    <script>
      // 添加li
      function add() {
        let liDom = document.createElement('li')
        liDom.textContent = '是兄弟，就是一起来玩贪玩蓝月'
        document.querySelector('#wrapper ul').appendChild(liDom)
        myScroll.refresh();
      }
      var myScroll = new IScrll('#wrapper')
    </script>
  </body>
</html>
```

iScroll结合Vue使用

实例化iScroll时，需要保证滚动内容已经渲染完成

滚动内容高度有变化时，需要调用refresh通知iscroll

```
<!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>
  <style>
    #wrapper {
      width: 400px;
      height: 200px;
      border: 1px solid green;
    }
  </style>
</head>

```

```
<body>
  <div id="app">
    <button @click="add">添加</button>
    <div id="wrapper">
      <ul>
        <li v-for="item in arr">{{item}}</li>
      </ul>
    </div>
  </div>
  <script src="./lib/vue.js"></script>
  <script src="./lib/iscroll.js"></script>
  <script>
    var myScroll = null
    const app = new Vue({
      el: "#app",
      data: {
        arr: [
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静'
        ]
      },
      methods: {
        add() {
          this.arr.push('台风前的宁静')
          myScroll.refresh()
        }
      },
    });
    let dom = document.getElementById('wrapper')
    myScroll = new IScroll(dom);
  </script>
</body>
```

```
60
61  </html>
```

ref与$refs属性

ref属性传送门

$refs属性传送门

Vue官方推荐获取dom的方式

使用

1. 标记dom 标签上ref="dom别名"
2. js里获取到dom vm.$refs.dom别名

如果两个标签的ref值是一样的，后者会覆盖前面的。

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-
   scale=1.0" />
6      <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7      <title>Document</title>
8    </head>
9    <body>
10     <div id="app">
11       <h2 ref='txt'>这是一个寂寞的天</h2>
12       <ul ref="list">
13         <li>啦啦啦</li>
14         <li>biu biu</li>
15       </ul>
16       <h2 ref="txt">下着有些伤心的雨</h2>
17       <button @click="print">打印</button>
18     </div>
19     <script src="./lib/vue.js"></script>
20     <script>
```

```
      const app = new Vue({
        el: "#app",
        data: {},
        methods: {
          print(){
            // 相同ref的两个元素，后面会覆盖前面的
            console.log(this.$refs.txt)
            console.log(this.$refs.list)
          }
        },
      });
      </script>
  </body>
</html>
```

Vue生命周期钩子 - mounted

最早能在这里获取到DOM

只会执行一次。

```
<!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">
    <h2 ref="txt">{{message}}</h2>
  </div>
  <script src="./lib/vue.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
```

```
      data: {
        message: '这是一个寂寞的天'
      },
      beforeCreate() {
        console.log(this.$refs.txt)
      },
      created() {
        console.log(this.$refs.txt)
      },
      beforeMount() {
        console.log(this.$refs.txt)
      },
      mounted() {
        // 最早能在这里获取到dom
        console.log(this.$refs.txt)
      }
    });
  </script>
</body>

</html>
```

iScroll结合Vue优化

1. iScroll的实例化放在mounted钩子函数里面，保证dom解析完全
2. ref和$refs获取DOM
3. 增加数据后，在nextTick里面iscroll refresh
4. 无须在DOM里面使用变量，最好直接添加为Vue实例的属性

```
<!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>
  <style>
```

```
    #wrapper {
      width: 400px;
      height: 200px;
      border: 1px solid green;
    }
  </style>
</head>

<body>
  <div id="app">
    <button @click="add">添加</button>
    <div id="wrapper" ref="wrapper">
      <ul>
        <li v-for="item in arr">{{item}}</li>
      </ul>
    </div>
  </div>
  <script src="./lib/vue.js"></script>
  <script src="./lib/iscroll.js"></script>
  <script>
    const app = new Vue({
      el: "#app",
      data: {
        arr: [
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静',
          '台风前的宁静'
        ],
        // myScroll: null
      },
      methods: {
        add() {
          this.arr.push('台风前的宁静')
```

```
            this.$nextTick(() => {
              this.myScroll.refresh()
            })
          }
        },
        mounted() {
          this.myScroll = new IScroll(this.$refs.wrapper);
        },
      });
    </script>
</body>

</html>

const app = {
  message: '这是一个寂寞的天'
}

app.username = 'joven'
console.log(app.message, app.username)
```

Demo-播放器结合iscroll

1. 引入iscroll
2. 三层嵌套结构
3. 实例化IScroll
    1. 在mounted里面实例化
    2. ref与$refs获取dom
4. 数据长度有变化时，需要refresh
    1. nextTick里面去refresh

Vue组件基本概念

传送门

组件就是一个独立的功能模块，包括html，js,css。

一次注册，到处使用。

Vue组件基本使用

1. 注册组件 `Vue.component(组件名,{template})`
    1. template是组件的dom结构
        1. dom结构必须有根元素包裹
        2. template:'html字符串' 不推荐
        3. 推荐用定义模板的方式

```
<script type="text/x-template" id="模板id">
template:'模板id'
```

2. 组件名在Vue的dom中当标签使用。可以使用N次。

方法和数据绑定

组件也是Vue实例

1. 组件里面的方法和数据绑定和实例化Vue是一样
2. 组件里面template属性是组件的dom结构
3. 组件里面data，必须是一个function

组件的data必须是一个函数

传送门

每个实例可以维护一份被返回对象的独立的拷贝

函数返回对象相当于重新创建一个对象

```

// 对象的方式
let data1 = {
  a: 1,
  b: 2
}
```

```
7
8  //组件实例化
9  let com1 = data1
10 let com2 = data1
11
12 com1.a = 666
13 // 对象的方式，组件的数据共享
14 console.log(com1, com2)
15
16 // 函数返回对象方式
17 function data2() {
18   return {
19     a: 1,
20     b: 2
21   }
22 }
23
24 //组件实例化
25 let com3 = data2()
26 let com4 = data2()
27 com3.a = 777
28 // 函数返回对象的方式，组件的数据独立
29 console.log(com3, com4)
```