

Vue学习第五天

反馈

回顾

1. 组件基本使用

1. 结构 template
2. data(){return{}}
3. methods:和之前一样
4. 除了1和2其他的都和之前的写法一样

```
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     <style>
9       .wrapper {
10         border: 1px solid #666;
11       }
12     </style>
13   </head>
14   <body>
15     <div id="app">
16       <counter></counter>
17       <counter></counter>
18     </div>
19     <script src="../lib/vue.js"></script>
20     <script type="text/x-template" id="counter_tpl">
21       <div class="wrapper">
22         <h2>点击了{{num}}次</h2>
23         <button @click="add">+</button>
24       </div>
25     </script>
26     <script>
27       vue.component('counter', {
28         template: '#counter_tpl',
29         data: () => {
30           return {
31             num: 0
32           }
33         },
34         methods: {
35           add() {
```

```

36         this.num++
37     }
38 }
39 })
40 const app = new Vue({
41   el: '#app',
42   data: {}
43 })
44 </script>
45 </body>
46 </html>

```

2. vue-router:

1. 高级的tab

2. 布局中

1. tab-nav: router-link
2. tab-content:router-view

```

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      <h1>Hello App!</h1>
12      <p>
13        <!-- 使用 router-link 组件来导航. -->
14        <!-- 通过传入 `to` 属性指定链接. -->
15        <!-- <router-link> 默认会被渲染成一个 `<a>` 标签 -->
16        <router-link to="/music1">歌曲1</router-link>
17        <router-link to="/bar">歌曲2</router-link>
18        <router-link to="/run">跑步</router-link>
19      </p>
20      <!-- 路由出口 -->
21      <!-- 路由匹配到的组件将渲染在这里 -->
22      <router-view></router-view>
23    </div>
24
25    <script src="https://unpkg.com/vue/dist/vue.js"></script>
26    <script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
27    <script>
28      // 0. 如果使用模块化机制编程，导入Vue和VueRouter，要调用 Vue.use(VueRouter)
29
30      // 1. 定义（路由）组件。
31      // 可以从其他文件 import 进来
32      const Foo = { template: '<div>foo</div>' }
33      const Bar = { template: '<div>bar</div>' }

```

```

34     const Run = { template: '<div>kakaka</div>' }
35
36     // 2. 定义路由
37     // 每个路由应该映射一个组件。 其中"component" 可以是
38     // 通过 vue.extend() 创建的组件构造器，
39     // 或者，只是一个组件配置对象。
40     // 我们晚点再讨论嵌套路由。
41     const routes = [{ path: '/music1', component: Foo }, { path: '/bar', component:
Bar }, {
42         path: '/run', component: Run
43     }]
44
45     // 3. 创建 router 实例，然后传 `routes` 配置
46     // 你还可以传别的配置参数，不过先这么简单着吧。
47     const router = new VueRouter({
48         routes // (缩写) 相当于 routes: routes
49     })
50
51     // 4. 创建和挂载根实例。
52     // 记得要通过 router 配置参数注入路由，
53     // 从而让整个应用都有路由功能
54     const app = new Vue({
55         router
56     }).$mount('#app')
57
58     // 现在，应用已经启动了！
59     </script>
60 </body>
61 </html>
62

```

1. 回顾路由，上代码
2. 定义组件，定义路由和组件的映射规则，创建router实例,把规则告诉router，挂载,把router传参Vue例
3. 使用router-link相当标签栏，router-view相当于标签下的内容, router-link to和路由规则要匹配

Demo-高级播放器-路由整合

实现步骤

1. 整合路由
 1. tab切换使用的是vue-router(路由)
 2. 显示的是组件
 3. 整合 多个页面，抽取为组件，使用路由的方式维护显示和隐藏
 4. git提交
2. 整合组件
 1. 把index隔壁的那些文件，结构，样式全部都整到index.html
 2. 抽取为组件
3. 为了晚上看代码简洁，删除了除index.html之外所有文件

路由高亮样式

传送门

1. vue-router 在我们切换 router-link 时，默认会自动的添加移除一个高亮的类名，
2. 如果需要更改通过 active-class 把这个属性设置给 router-link 即可实现使用自己的类名作为高亮类名

```
1 | <router-link active-class='active'>导航</router-link>
```

编程式导航

传送门

1. 编程式导航的本质是

```
1 | router.push('地址')
```

2. 声明式导航的本质是

```
1 | <router-link to='/run'>去跑步</router-link>
```

3. 适用情景

1. 声明式导航：点了就跳转，没有任何逻辑 类似于(a标签设置了href)。可以由编程式实现
2. 编程式导航：跳转的同时有其他逻辑需要执行

```
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 |      <!-- tab - nav 导航 声明式导航 -->
12 |      <router-link to="/music1">歌曲1</router-link>
13 |      <router-link to="/bar">歌曲2</router-link>
14 |      <router-link to="/run">边唱歌鞭炮</router-link>
15 |      <h2>编程式导航</h2>
16 |      <input type="button" value="点我去唱歌" @click="toSing">
17 |
18 |      <!-- tab - content -->
19 |      <router-view></router-view>
20 |    </div>
21 |  </body>
```

```

22 </html>
23 <!-- 放上面会影响页面的接在 js加载完毕之后页面是看不到的 -->
24 <script src="https://unpkg.com/vue/dist/vue.js"></script>
25 <script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
26 <script>
27   // 1. 定义组件 简化的写法
28   const Foo = { template: '<div>foo</div>' }
29   const Bar = { template: '<div>bar</div>' }
30   const run = { template: '<div>咋咋咋！！！！的跑！！！！</div>' }
31
32   // 2. 定义规则
33   // url和组件的对应关系
34   // const routes = [
35   const routers = [
36     { path: '/music1', component: Foo },
37     { path: '/bar', component: Bar },
38     { path: '/run', component: run }
39   ]
40
41   // 3. 把路由规则 设置给路由对象
42   const router = new VueRouter({
43     // routes // (缩写) 相当于 routes: routes
44     routes: routers // routers: routers
45   })
46
47   // 4. 创建和挂载根实例。
48
49   const app = new Vue({
50     el: '#app',
51     methods: {
52       // 跳转去唱歌
53       toSing(){
54         // console.log('唱歌去')
55         // 用路由对象跳转
56         // router.push('/bar')
57         router.push('/niubility')
58       }
59     },
60     router // router:router
61   })
62
63   // 现在，应用已经启动了！
64 </script>
65

```

动态路由匹配

[传送门](#)

1. 修改路由规则 `/user => /user/:key`

2. `:key` 是一个占位，名字可以更改
3. 切换路由是，地址 `/user => /user/数据`
4. 携带者数据切换了路由之后，如何获取数据

1. `data` 中会自动的被添加上一个 `$route` 内部的 `params` 就保存了我们传递的数据

2.

```
▼ data
  ▼ $route
    path: "/bar/西兰花"
    ► query: Object (empty)
    ▼ params: Object
      food: "西兰花"
      fullPath: "/bar/西兰花"
    ► meta: Object (empty)
```

```
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      <h1>Hello App!</h1>
12      <p>
13        <router-link to="/music1">歌曲1</router-link>
14        <router-link to="/bar/Joven">歌曲2</router-link>
15      </p>
16      <router-view></router-view>
17    </div>
18
19    <script src="https://unpkg.com/vue/dist/vue.js"></script>
20    <script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
21    <script>
22      const Foo = { template: '<div>foo</div>' }
23      const Bar = { template: '<div>{{$route.params.name}}</div>' }
24
25      const routes = [{ path: '/music1', component: Foo }, { path: '/bar/:name',
component: Bar }]
26      const router = new VueRouter({
27        routes
28      })
29
30      const app = new Vue({
31        router,
32        methods: {
33          toSing() {
34            router.push('music1')
```

```
35     }  
36   }  
37   }).$mount('#app')  
38 </script>  
39 </body>  
40 </html>
```

Demo-歌曲搜索

黑云音乐

实现步骤

1. 顶部的搜索框双向数据绑定v-model.trim :search
2. 点击搜索或者点击回车
 1. @click @keyup.enter :searchMusic
 2. 把搜索的内容，传递给 搜索组件
 1. 动态路由匹配
 2. router.push('/result/\${this.search}')
 3. 同时让搜索组件显示出来
 1. 编程式导航
 2. 修改路由规则 `/result=>/result/:search`

注意点

1. 路由切换时需要携带数据，用的是 动态路由匹配
2. 代码的方式跳转，用的是 编程式导航

生命周期钩子 -created

1. beforeCreate:
 1. Vue实例被创建，但是传入的参数还没有设置上去
2. created
 1. Vue实例被创建，传入的参数已经设置给这个Vue实例了
 2. 如果要操纵数据最起码在这个里面
3. beforeMount
 1. 开始解析结构，把数据和解构关联起来，刚刚开始，页面上看不到
4. mounted

1. 数据和页面已经关联起来，这一步dom元素已经可以获取到

```
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"></div>
11  </body>
12 </html>
13 <script src="../lib/vue.js"></script>
14
15 <script>
16   const app = new Vue({
17     el: '#app',
18     data: {
19       secret: '我有一个小秘密，就不告诉你！'
20     },
21     beforeCreate() {
22       console.log('beforeCreate')
23       console.log(this.secret)
24     },
25     created() {
26       console.log('created')
27       console.log(this.secret)
28     }
29   })
30 </script>
```

Demo-高级播放器-结果搜索

实现步骤

1. 当 result 组件创建出来之后（出现）之后，
 1. 使用生命周期钩子（自动执行）created
 2. 尽可能早一些执行的，让用户早一些看到数据
2. 获取传递过来的关键字 `this.$route.params.键`
3. 通过关键字调用接口，axios
4. 数据获取到之后，渲染到页面上
 1. then
 2. v-for :musicList

过滤器基本使用

过滤器

Vue.js 允许你自定义过滤器，可被用于一些常见的文本格式化。过滤器可以用在两个地方：**双花括号插值**和**v-bind 表达式** (后者从 2.1.0+ 开始支持)。

1. 定义的方式

1. vue中filters:{}

2. 一个过滤器一个方法

3. 使用的使用 {{ 数据 | 过滤器 }}

4. 过滤器需要接收一个参数，参数就是处理的数据

5. 内部处理完毕之后

6. return 出来页面会显示 返回出来的那个值

7. 不会修改原始值

```
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>公历生日:{{formatDate1}}</h2>
12      <h2>农历生日:{{formatDate2}}</h2>
13      <h2>过滤器的方式</h2>
14      <h2>公历生日:{{date1|formatDate}}</h2>
15      <h2>农历生日:{{date2|formatDate}}</h2>
16    </div>
17    <script src="./lib/vue.js"></script>
18    <script src="./lib/moment.js"></script>
19    <script>
20      const app = new Vue({
21        el: "#app",
22        data: {
23          date1: '2019-8-13',
24          date2: '2019-7-13'
25        },
26        computed: {
27          formatDate1(){
28            return moment(this.date1).format('YYYY.MM.DD')
29          },
30          formatDate2(){
```

```

32         return moment(this.date2).format('YYYY.MM.DD')
33     }
34 },
35     filters:{
36         formatDate(date){
37             return moment(date).format('YYYY.MM.DD')
38         }
39     }
40 });
41 </script>
42 </body>
43 </html>

```

Demo-过滤器处理result中搜索的结果

歌曲	歌手	时长
周兴哲	《爱，教会我们的事》	287306

实现步骤

1. 处理歌手名


1. 为了防止多人演唱只显示第一个的问题
2. 添加过滤器 处理歌手
 1. filters:{ formatSinger(arr) }
 2. {{ item.artists | formatSinger }}
3. 过滤器内部逻辑

1. 循环数组，拼接名字 用 '/'
2. 移除最后的那个 '/'
3. return

2. 处理时间

1. 添加过滤器 处理 时间
 1. filters:{ formatTime(time) }
 2. {{item.duration | formatTime }}
2. 处理逻辑
 1. 毫秒->秒
 2. 算出分 60的整数倍 除
 3. 剩余的部分作为秒 取余

注意点

1. 时间从毫秒转为 时分秒，
 1. 先除 再取余
2. 过滤器的特点是，不修改数据的情况下 更改数据的显示效果
3. 过滤器的使用 

1. 这个| 也叫 管道符

Demo-点击mv 播放MV

实现步骤

1. result组件中 生成mv按钮时，绑定点击事件 携带mvid 跳转到mv路由那

1. router.push('/mv/mv的id')
2. 路由规则 /mv -> /mv/:mvid

2. mv组件中

1. 获取mvid
2. axios接口调用
3. 数据回来之后，渲染到页面上
 1. 歌名：songName
 2. 歌手名:singerName
 3. mv的地址:mvUrl

重点

实现步骤和 搜歌 类似 跳转，携带数据

预习

单文件组件 xxx.vue

[脚手架](#)

[安装](#)

[创建项目](#)