

# Vue 核心技术与实战

## day06



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌

## 声明式导航 - 导航链接

需求：实现导航高亮效果

发现音乐	我的音乐	朋友
发现音乐		
发现音乐		
发现音乐		
发现音乐		

## 声明式导航 - 导航链接

需求：实现导航高亮效果

vue-router 提供了一个全局组件 router-link (取代 a 标签)

- ① 能跳转，配置 to 属性指定路径(必须)。本质还是 a 标签，to 无需 #
- ② 能高亮，默认就会提供高亮类名，可以直接设置高亮样式



```
<template>
  <div>
    <div class="footer_wrap">
      <router-link to="/find">发现音乐</router-link>
      <router-link to="/my">我的音乐</router-link>
      <router-link to="/part">朋友</router-link>
    </div>
    <div class="top">
      <router-view></router-view>
    </div>
  </div>
</template>
```

```
<template>
  <div>
    <div class="footer_wrap">
      <a href="#/find">发现音乐</a>
      <a href="#/my">我的音乐</a>
      <a href="#/part">朋友</a>
    </div>
    <div class="top">
      <router-view></router-view>
    </div>
  </div>
</template>
```



## 小结

### 1. router-link是什么?

vue-router提供的全局组件, 用于替换 a 标签

### 2. router-link怎么用?

```
<router-link to="/路径值" ></router-link>
```

必须传入to属性, 指定路由路径值

### 3. router-link好处?

能跳转, 能高亮 (自带激活时的类名)

## 声明式导航 - 两个类名

说明：我们发现 router-link 自动给当前导航添加了 **两个高亮类名**

```
<div class="footer_wrap">
  <a href="#/find" class>发现音乐</a>
  <a href="#/my" aria-current="page" class="router-link-exact-active router-link-active">我的音乐</a> == $0
  <a href="#/friend" class>朋友</a>
</div>
```

① router-link-active **模糊匹配 (用的多)**

to="/my" 可以匹配 /my /my/a /my/b ....

② router-link-exact-active **精确匹配**

to="/my" 仅可以匹配 /my





## 小结

router-link 会自动给当前导航添加两个类名，有什么区别呢？

router-link-active 模糊匹配 (用的多)

router-link-exact-active 精确匹配

## 声明式导航 - 两个类名

说明：router-link 的 两个高亮类名 太长了，我们希望能定制怎么办？

```
<div class="footer_wrap">
  <a href="#/find" class>发现音乐</a>
  <a href="#/my" aria-current="page" class="router-link-exact-active router-link-active">我的音乐</a> == $0
  <a href="#/friend" class>朋友</a>
</div>
```

```
const router = new VueRouter({
  routes: [...],
  linkActiveClass: "类名1",
  linkExactActiveClass: "类名2"
})
```

```
<div class="footer_wrap"> == $0
  <a href="#/find" class>发现音乐</a>
  <a href="#/my" aria-current="page" class="类名2 类名1">我的音乐</a>
  <a href="#/friend" class>朋友</a>
</div>
```



## 小结

如何自定义 router-link 的 **两个高亮类名**？

linkActiveClass 模糊匹配 类名自定义

linkExactActiveClass 精确匹配 类名自定义



## 声明式导航 - 跳转传参

目标：在跳转路由时, 进行传值

1. 查询参数传参
2. 动态路由传参



## 声明式导航 - 跳转传参

目标：在跳转路由时，进行传值

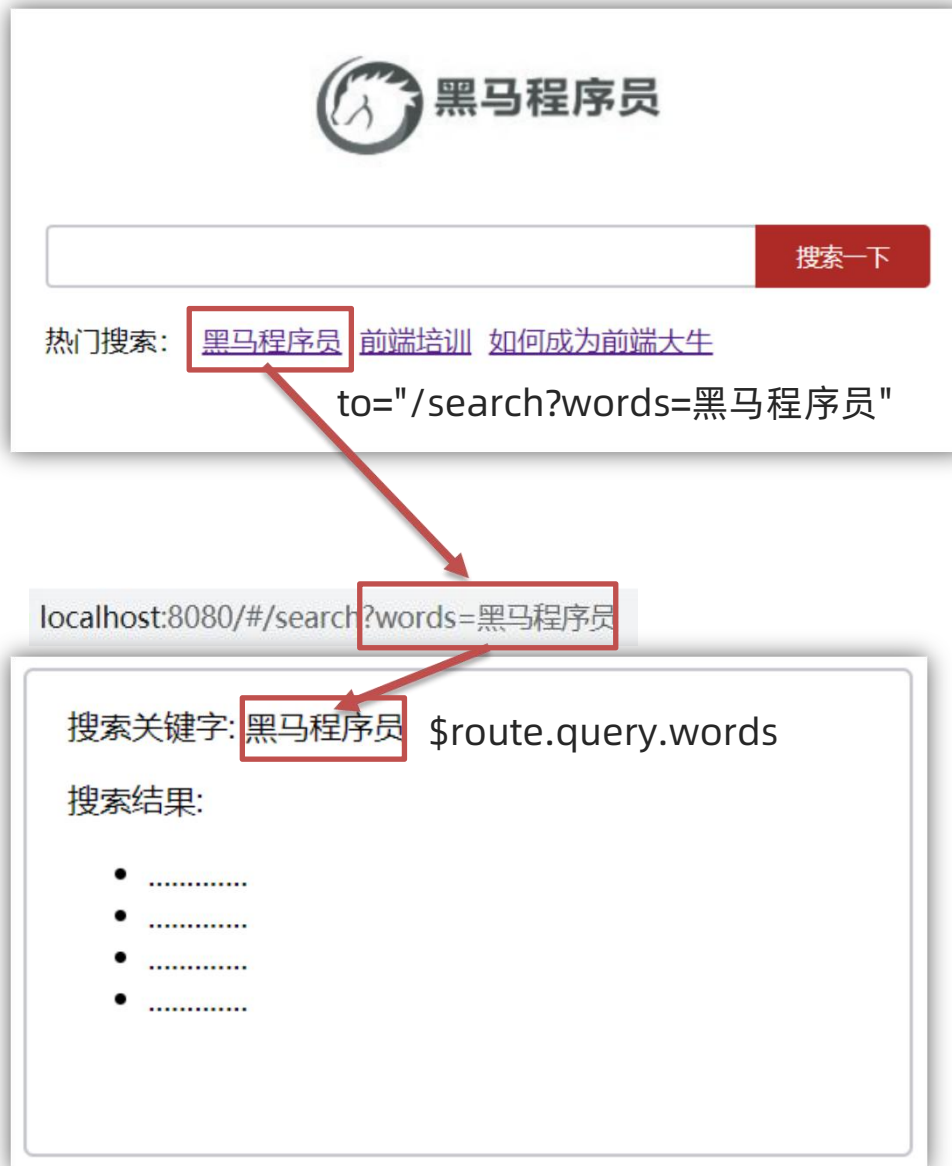
### 1. 查询参数传参

① 语法格式如下

- `to="/path?参数名=值"`

② 对应页面组件接收传递过来的值

- `$route.query.参数名`



黑马程序员

搜索一下

热门搜索: 黑马程序员 前端培训 如何成为前端大生

`to="/search?words=黑马程序员"`

localhost:8080/#/search?words=黑马程序员

搜索关键字: 黑马程序员 `$route.query.words`

搜索结果:

- .....
- .....
- .....
- .....

## 声明式导航 - 跳转传参

目标：在跳转路由时, 进行传值

### 2. 动态路由传参

#### ① 配置动态路由

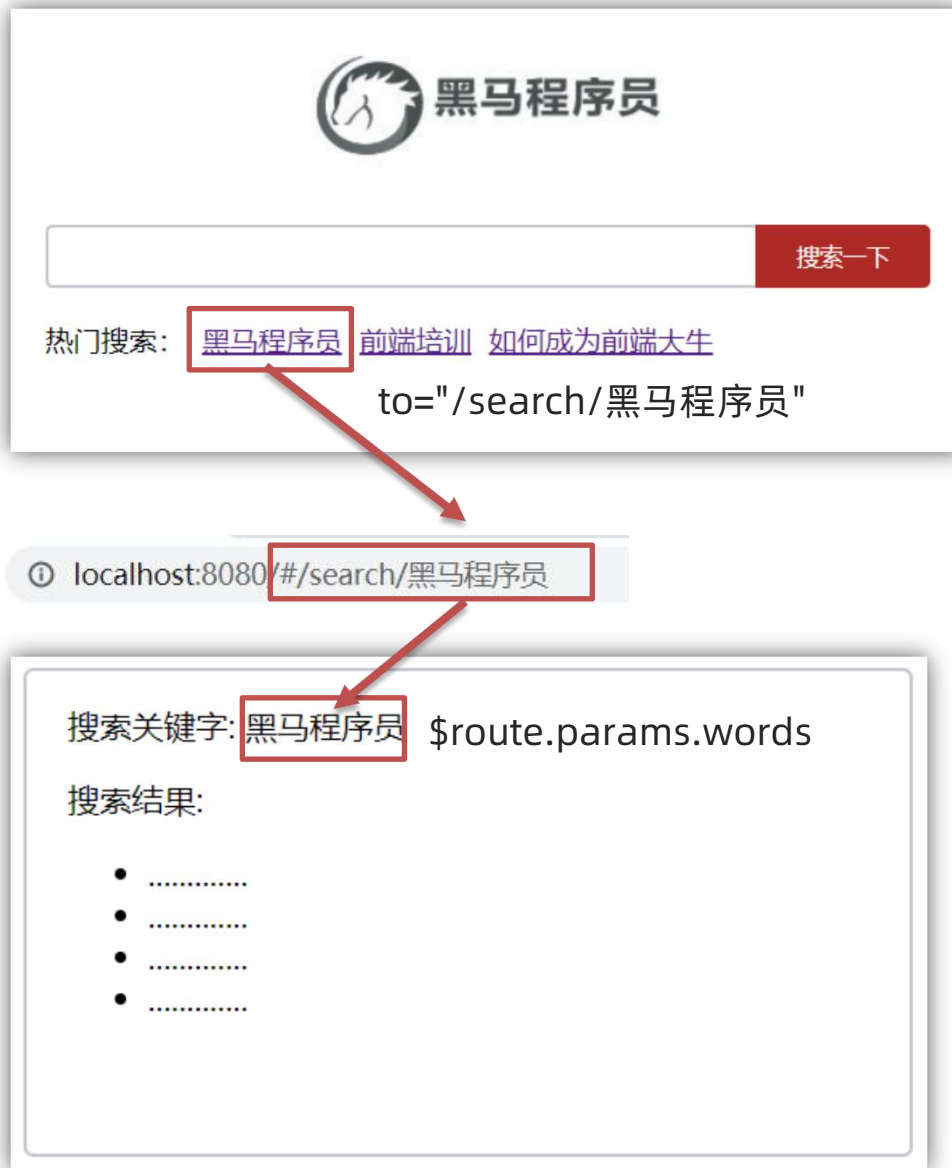
```
const router = new VueRouter({
  routes: [
    ...,
    {
      path: '/search/:words',
      component: Search
    }
  ]
})
```

#### ② 配置导航链接

- to="/path/参数值"

#### ③ 对应页面组件接收传递过来的值

- \$route.params.参数名



The diagram illustrates the flow of data from a search input to a route configuration and finally to a component. It consists of three main parts:

- Top Panel (Search Interface):** Shows a search bar with the text "黑马程序员" (Hei Ma Cheng Xu Yuan) entered. Below the search bar, the "热门搜索:" (Hot Search) section lists "黑马程序员", "前端培训", and "如何成为前端大牛". The "to="/search/黑马程序员" is shown as the resulting path.
- Middle Panel (URL):** Shows the browser address bar with the URL "localhost:8080/#/search/黑马程序员". The path part of the URL is highlighted with a red box.
- Bottom Panel (Search Results):** Shows the search results page. The "搜索关键字:" (Search Keyword) is "黑马程序员", and the "搜索结果:" (Search Results) section is empty. The text "\$route.params.words" is shown next to the keyword, indicating the parameter passed from the URL.

Red arrows indicate the flow of data: from the search input to the URL, and from the URL to the search results page.

## 声明式导航 - 跳转传参

两种传参方式的区别

### 1. 查询参数传参 (比较适合传多个参数)

① 跳转: `to="/path?参数名=值&参数名2=值"`

② 获取: `$route.query.参数名`

### 2. 动态路由传参 (优雅简洁, 传单个参数比较方便)

① 配置动态路由: `path: "/path/参数名"`

② 跳转: `to="/path/参数值"`

③ 获取: `$route.params.参数名`

声明式导航跳转时, 有几种方式传值给路由页面?



## 小结

### ① 查询参数传参 (多个参数)

跳转: `to="/path?参数名=值"`

接收: `$route.query.参数名`

### ② 动态路由传参 (简洁优雅)

路由: `/path/:参数名`

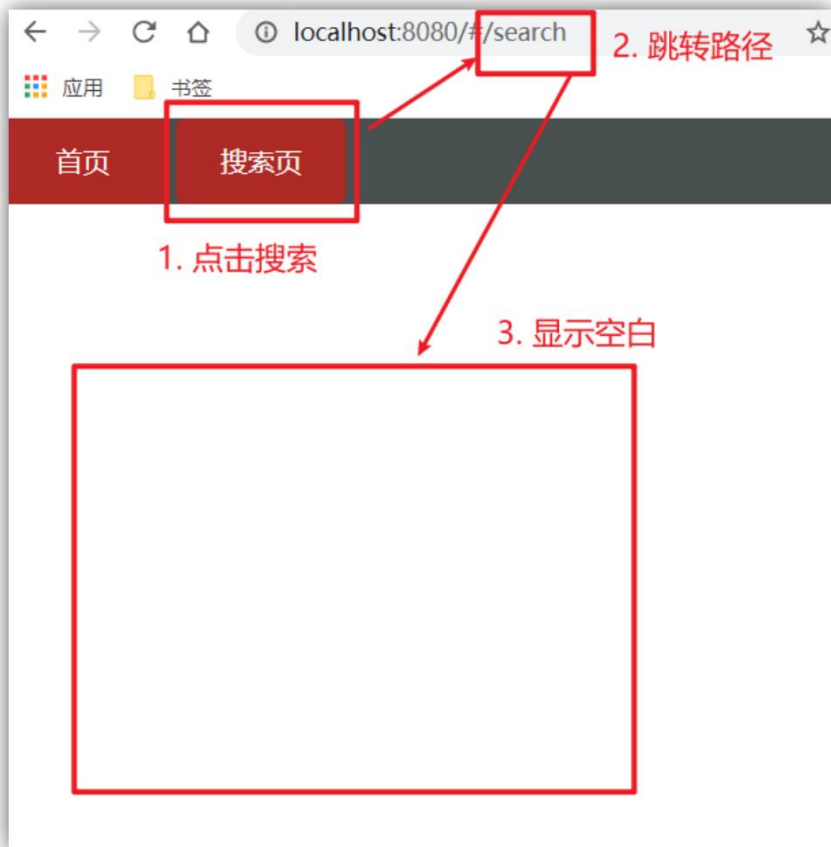
跳转: `to="/path/值"`

接收: `$route.params.参数名`

## 动态路由参数可选符

**问题：**配了路由 `path: "/search/:words"` 为什么按下面步骤操作，会未匹配到组件，显示空白？

**原因：**`/search/:words` 表示，必须要传参数。如果不传参数，也希望匹配，可以加个可选符 `"?"`



```
const router = new VueRouter({
  routes: [
    { path: '/', redirect: '/home' },
    { path: '/home', component: Home },
    { path: '/search/:words?', component: Search }
  ]
})
```

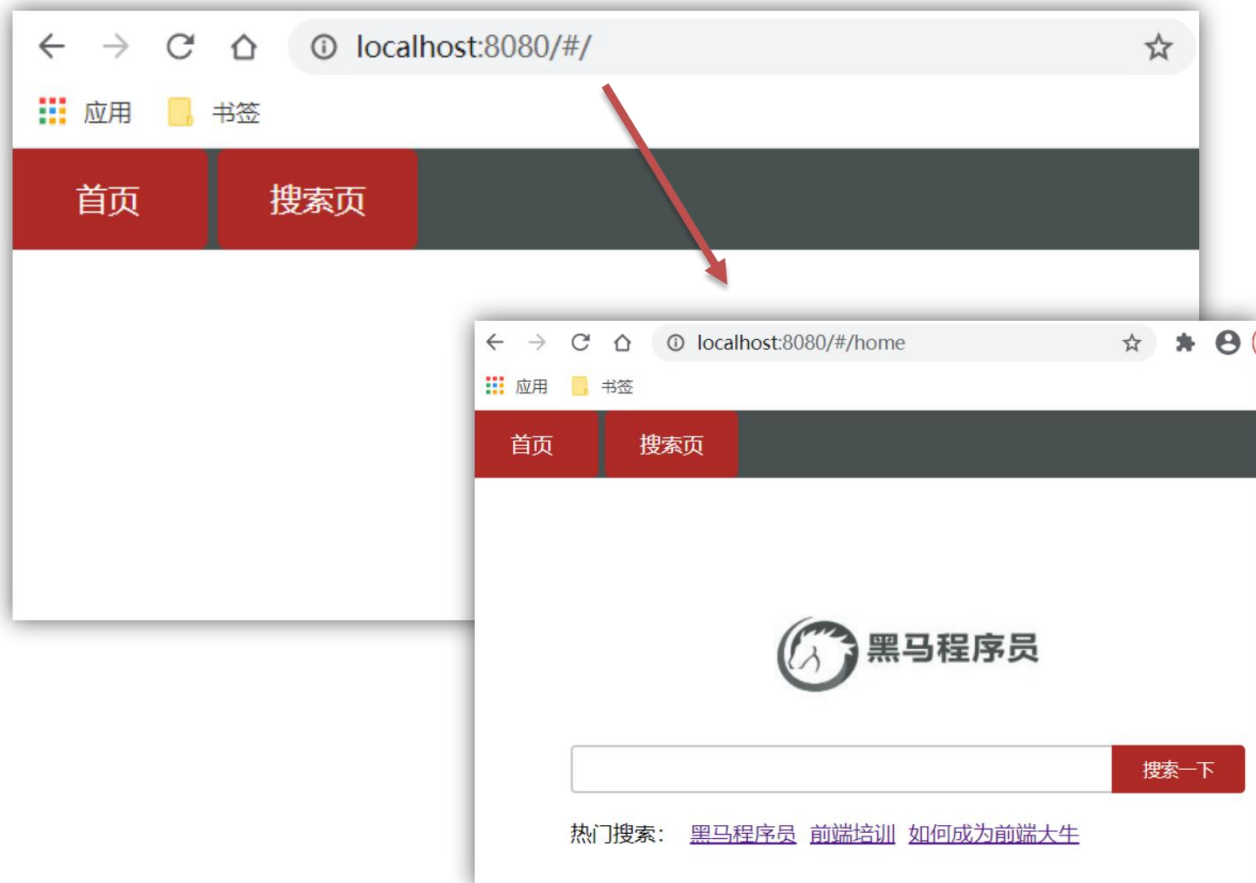
## Vue路由 - 重定向

**问题：**网页打开，url 默认是 / 路径，未匹配到组件时，会出现空白

**说明：**重定向 → 匹配path后，强制跳转path路径

**语法：** { path: 匹配路径, redirect: 重定向到的路径 },

```
const router = new VueRouter({
  routes: [
    { path: '/', redirect: '/home' },
    { path: '/home', component: Home },
    { path: '/search/:words', component: Search }
  ]
})
```



## Vue路由 - 404

**作用：**当路径找不到匹配时，给个提示页面

**位置：**配在路由最后

**语法：**path: "\*" (任意路径) - 前面不匹配就命中最后这个

**Page Not Find 404**

```
import NotFound from '@views/NotFound'

const router = new VueRouter({
  routes: [
    { path: '/', redirect: '/home' },
    { path: '/home', component: Home },
    { path: '/search/:words?', component: Search },
    { path: '*', component: NotFound }
  ]
})
```



## Vue路由 - 模式设置

问题：路由的路径看起来不自然，有#，能否切成真正路径形式？

- hash路由(默认) 例如：http://localhost:8080/#/home
- history路由(常用) 例如：http://localhost:8080/home (以后上线需要服务器端支持)

```
const router = new VueRouter({  
  routes,  
  mode: "history"  
})
```

## 程式导航 - 基本跳转

问题：点击按钮跳转如何实现？

程式导航：用JS代码来进行跳转

两种语法：

① path 路径跳转

② name 命名路由跳转



## 程式导航 - 基本跳转

问题：点击按钮跳转如何实现？

程式导航：用JS代码来进行跳转

两种语法：

### ① path 路径跳转 (简易方便)

```
this.$router.push('路由路径')  
  
this.$router.push({  
  path: '路由路径'  
})
```



## 程式导航 - 基本跳转

问题：点击按钮跳转如何实现？

程式导航：用JS代码来进行跳转

两种语法：

① path 路径跳转

② name 命名路由跳转 (适合 path 路径长的场景)

```
this.$router.push({  
  name: '路由名'  
})
```

```
{ name: '路由名', path: '/path/xxx', component: XXX },
```



编程式导航有几种跳转方式？

① 通过路径跳转 (简易方便)

```
this.$router.push('路由路径')  
  
this.$router.push({  
  path: '路由路径'  
})
```

② 通过路由名字跳转 (适合路径名字长的场景)

```
this.$router.push({  
  name: '路由名'  
})
```

```
{ name: '路由名', path: '/path/xxx', ... },
```



小结

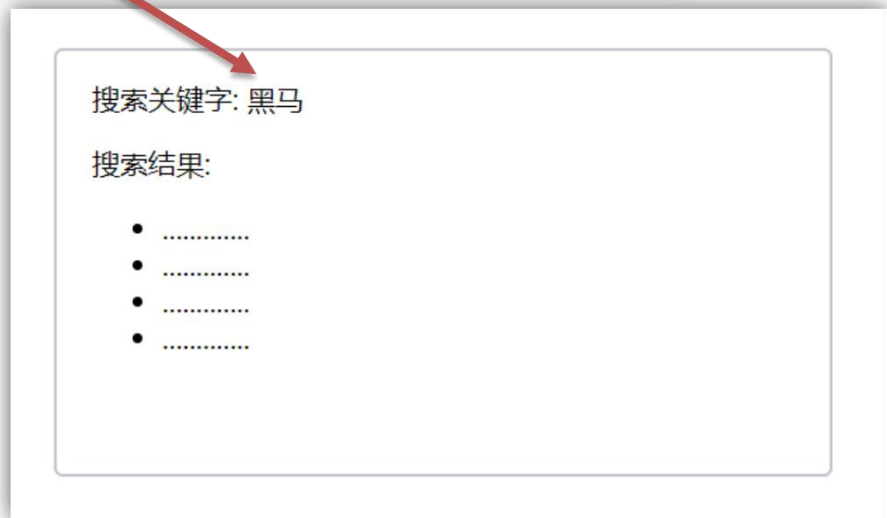
## 程式导航 - 路由传参

问题：点击搜索按钮，跳转需要传参如何实现？

两种传参方式：查询参数 + 动态路由传参

两种跳转方式，对于两种传参方式都支持：

- ① path 路径跳转传参
- ② name 命名路由跳转传参



## 程式导航 - 路由传参

问题：点击搜索按钮，跳转需要传参如何实现？

两种传参方式：查询参数 + 动态路由传参

两种跳转方式，对于两种传参方式都支持：

① path 路径跳转传参 (query传参)

② name 命名路由跳转传参

```
this.$router.push('/路径?参数名1=参数值1&参数2=参数值2')
this.$router.push({
  path: '/路径',
  query: {
    参数名1: '参数值1',
    参数名2: '参数值2'
  }
})
```



搜索关键字: 黑马 `$route.query.参数名`

搜索结果:

- .....
- .....
- .....
- .....

## 程式导航 - 路由传参

问题：点击搜索按钮，跳转需要传参如何实现？

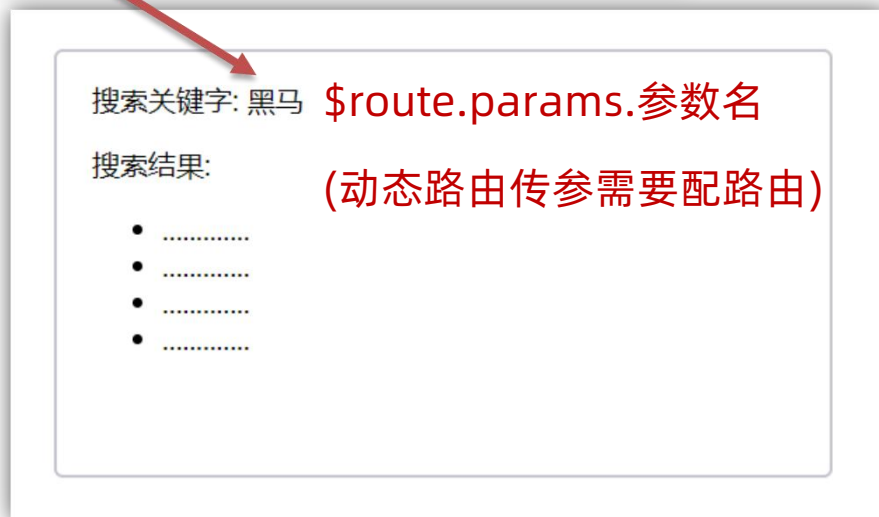
两种传参方式：查询参数 + 动态路由传参

两种跳转方式，对于两种传参方式都支持：

① path 路径跳转传参 (动态路由传参)

② name 命名路由跳转传参

```
this.$router.push('/路径/参数值')  
this.$router.push({  
  path: '/路径/参数值'  
})
```





## 程式导航 - 路由传参

问题：点击搜索按钮，跳转需要传参如何实现？

两种传参方式：查询参数 + 动态路由传参

两种跳转方式，对于两种传参方式都支持：

① path 路径跳转传参

② name 命名路由跳转传参 (query传参)

```
this.$router.push({  
  name: '路由名字',  
  query: {  
    参数名1: '参数值1',  
    参数名2: '参数值2'  
  }  
})
```



搜索关键字: 黑马 `$route.query.参数名`

搜索结果:

- .....
- .....
- .....
- .....

## 程式导航 - 路由传参

问题：点击搜索按钮，跳转需要传参如何实现？

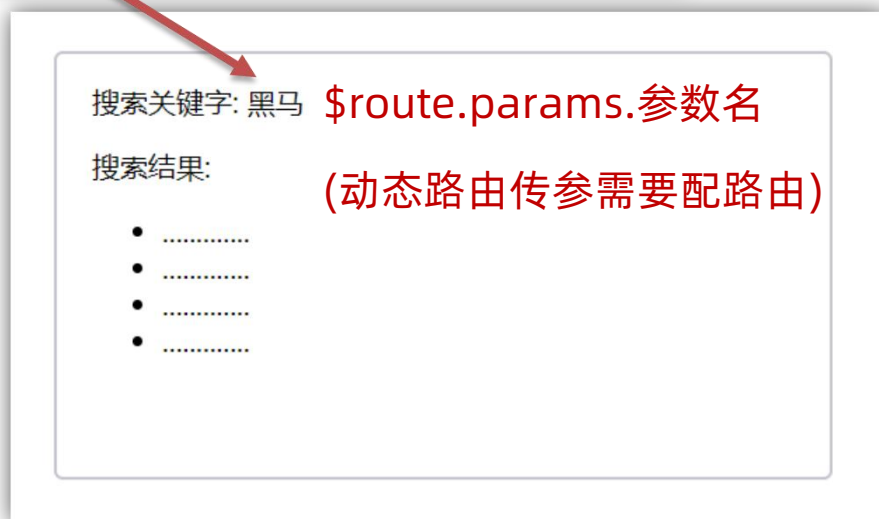
两种传参方式：查询参数 + 动态路由传参

两种跳转方式，对于两种传参方式都支持：

① path 路径跳转传参

② name 命名路由跳转传参 (动态路由传参)

```
this.$router.push({  
  name: '路由名字',  
  params: {  
    参数名: '参数值',  
  }  
})
```



程式导航，如何跳转传参？

## 1. path 路径跳转

### ① query传参

```
this.$router.push('/路径?参数名1=参数值1&参数2=参数值2')  
this.$router.push({  
  path: '/路径',  
  query: {  
    参数名1: '参数值1',  
    参数名2: '参数值2'  
  }  
})
```

### ② 动态路由传参 (需要配动态路由)

```
this.$router.push('/路径/参数值')  
this.$router.push({  
  path: '/路径/参数值'  
})
```



小结

程式导航，如何跳转传参？

## 2. name 命名路由跳转

### ① query传参

```
this.$router.push({  
  name: '路由名字',  
  query: {  
    参数名1: '参数值1',  
    参数名2: '参数值2'  
  }  
})
```

### ② 动态路由传参 (需要配动态路由)

```
this.$router.push({  
  name: '路由名字',  
  params: {  
    参数名: '参数值',  
  }  
})
```



小结

## 案例效果：



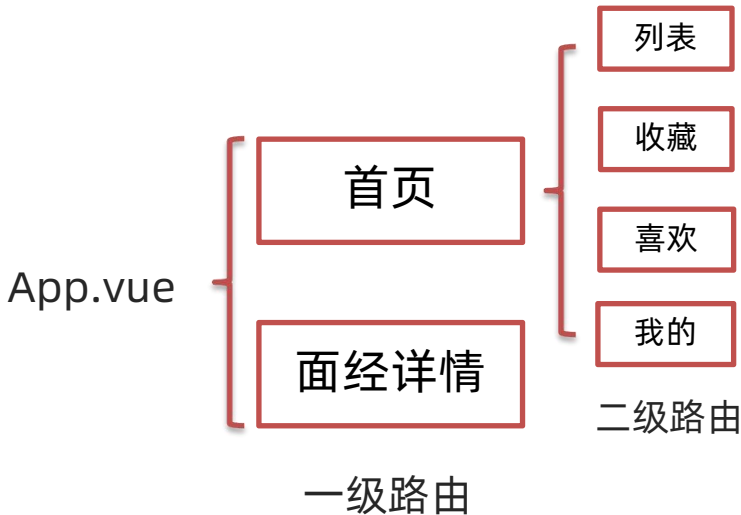
## 分析：配路由 + 实现功能

### 1. 配路由

- ① 首页 和 面经详情，两个一级路由
- ② 首页内嵌四个可切换页面 (嵌套二级路由)

### 2. 实现功能

- ① 首页请求渲染
- ② 跳转传参 到 详情页，详情页渲染
- ③ 组件缓存，优化性能

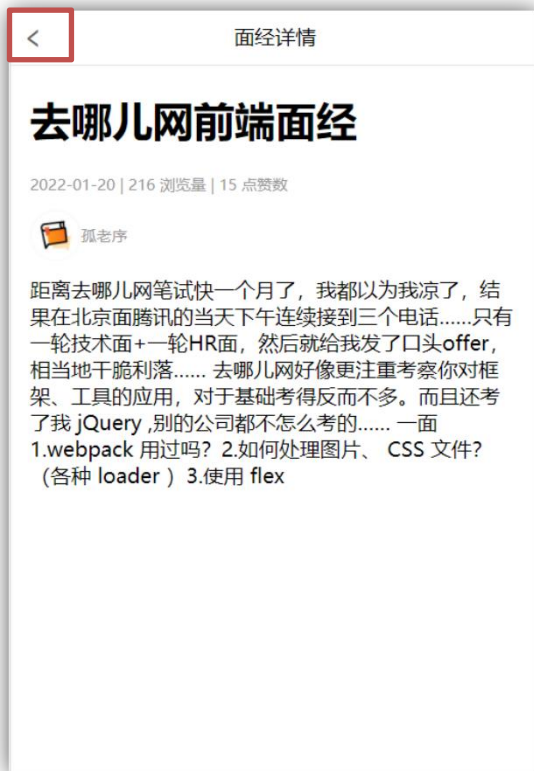


## 组件缓存 keep-alive

问题：从面经 点到 详情页，又点返回，数据重新加载了 → 希望回到原来的位置

原因：路由跳转后，组件被销毁了，返回回来组件又被重建了，所以数据重新被加载了

解决：利用 keep-alive 将组件缓存下来



## 组件缓存 keep-alive

### 1. keep-alive是什么

keep-alive 是 Vue 的内置组件，当它包裹动态组件时，会缓存不活动的组件实例，而不是销毁它们。

keep-alive 是一个抽象组件：它自身不会渲染成一个 DOM 元素，也不会出现在父组件链中。

### 2. keep-alive的优点

在组件切换过程中 把切换出去的组件保留在内存中，防止重复渲染DOM，

减少加载时间及性能消耗，提高用户体验性。

```
<template>
  <div class="h5-wrapper">
    <keep-alive>
      <router-view></router-view>
    </keep-alive>
  </div>
</template>
```

问题：缓存了所有被切换的组件

## 组件缓存 keep-alive

### 3. keep-alive的三个属性

- ① include : 组件名数组，只有匹配的组件会被缓存
- ② exclude : 组件名数组，任何匹配的组件都不会被缓存
- ③ max : 最多可以缓存多少组件实例

```
<template>
  <div class="h5-wrapper" :include="['LayoutPage']>
    <keep-alive >
      <router-view></router-view>
    </keep-alive>
  </div>
</template>
```



## 组件缓存 keep-alive

### 4. keep-alive的使用会触发两个生命周期函数

activated 当组件被激活（使用）的时候触发 → 进入这个页面的时候触发

deactivated 当组件不被使用的时候触发 → 离开这个页面的时候触发

组件缓存后就不会执行组件的created, mounted, destroyed 等钩子了

所以其提供了activated 和 deactivated钩子，帮我们实现业务需求。

```
activated () {  
  console.log('activated 激活 → 进入页面');  
},  
deactivated() {  
  console.log('deactivated 失活 → 离开页面');  
}
```



## 小结

### 1. keep-alive是什么

Vue 的内置组件，包裹动态组件时，可以缓存

### 2. keep-alive的优点

组件切换过程中 把切换出去的组件保留在内存中(提升性能)

### 3. keep-alive的三个属性 (了解)

① include : 组件名数组，只有匹配的组件会被缓存

② exclude : 组件名数组，任何匹配的组件都不会被缓存

③ max : 最多可以缓存多少组件实例

### 4. keep-alive的使用会触发两个生命周期函数 (了解)

activated 当组件被激活（使用）的时候触发 → 进入页面触发

deactivated 当组件不被使用的时候触发 → 离开页面触发



# 总结

1. 项目案例实现的基本步骤分哪两大步？

① 配路由 ② 实现页面功能

2. 嵌套路由的关键配置项是什么？

children

3. 路由传参两种方式？

① 查询参数传参，`$route.query.参数名` (适合多个参数)

② 动态路由传参，`$route.params.参数名` (更简洁直观)

4. 缓存组件可以用哪个内置组件？

keep-alive

三个属性：include exclude max

两个钩子：activated deactivated



传智教育旗下高端IT教育品牌