

Vue路由

脚本引入

引入的顺序是首先引入vue.js，然后是vue-router.js。

```
<script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
<script src="https://cdn.bootcss.com/vue-router/3.0.6/vue-router.min.js"></script>
```

创建路由

```
<body>
  <div class="main" id="app1">

  </div>
  <script>
    var router = new VueRouter({}); // 创建Vue路由实例
    new Vue({
      el: "#app1",
      data: {},
      router: router // 注入 路由
    })
  </script>
</body>
```

路由创建成功后，访问Vue页面时，URL地址后面将出现 #/



完整步骤与实例

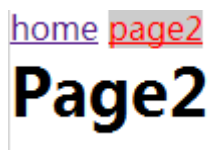
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Vue路由</title>
  <link rel="stylesheet" href="index.css">
  <!-- 0.引入 -->
  <script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
  <script src="https://cdn.bootcss.com/vue-router/3.0.6/vue-router.min.js"></script>
</head>
```

```

/*--8.当前超链接激活样式--*/
.active{
  color: red;
  background: #ccc;
}
</style>
</head>
<body>
  <div class="main" id="app1">
    <!-- 6.router-link 路由超链接 -->
    <router-link to="/home">home</router-link>
    <router-link to="/page2">page2</router-link>
    <!-- 5.router-view 路由对应的组件显示的位置 -->
    <router-view></router-view>
  </div>
  <script>
    //1. 定义组建
    var home = {template: "<h1>HOME</h1>"};
    var page2 = {template: "<h1>Page2</h1>"};
    //2.路由的映射 路径->组建
    var routes = [
      {path: "/home", component: home},
      {path: "/page2", component: page2}
    ];
    //3.创建路由
    var router = new VueRouter({
      routes: routes,
      linkActiveClass: "active", //7.设定当前超链接样式
    });
    //4.注入 路由
    new Vue({
      el: "#app1",
      data: {},
      router: router //注入 路由
    })
  </script>
</body>
</html>

```

效果如下：



home page2

Page2

路由映射的进一步配置

```
var routes = [
  {path: "/", component: home}, //9. 默认路径设置为home
  {path: "/home", component: home},
  {path: "/page2", component: page2},
  {path: "/*", redirect: "/home"}, //10. 不存在的路径重定向到home
];
```

另一种写法与程式导航

```
<!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>Vue路由</title>
  <link rel="stylesheet" href="index.css">
  <!-- 0.引入 -->
  <script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
  <script src="https://cdn.bootcss.com/vue-router/3.0.6/vue-router.min.js"></script>
  <style>
    /*--8. 当前超链接激活样式--*/
    .active{
      color: red;
      background: #ccc;
    }
  </style>
</head>
<body>
  <div class="main" id="app1">
    <router-view></router-view>
  </div>
  <!-- 1.template在标签中定义 -->
  <template id="home">
    <div>
      <h1>home1</h1>
      <button @click="goPage2">跳转到page2</button> <!-- 4. 设定Button单击事件 -->
    </div>

  </template>
  <template id="page2">
    <div>
      <h1>page2</h1>
      <button @click="goHome">跳转到Home</button>
    </div>
  </template>

  <script>
    //2. 组件的template直接指向<template>的ID
    var home = {
      template: "#home",
```

```

        methods:{
            goPage2(){//5.在home组件中响应button的单击事件
                this.$router.push("/page2")
            }
        }
    };
    var page2 = {
        template: "#page2",
        methods: {
            goHome(){//5.在home组件中响应button的单击事件
                this.$router.push("/home")
            }
        }
    };
    //3.routes映射直接写在router定义里
    var router = new VueRouter({
        routes: [
            {path: "/", component: home},
            {path: "/home", component: home},
            {path: "/page2", component: page2},
            {path: "/*", redirect: "/home"},
        ],
        linkActiveClass: "active",
    });

    new Vue({
        el: "#app1",
        data: {},
        router: router//注入 路由
    })
</script>
</body>
</html>

```

效果：

通过单击按钮可以实现路径跳转。

home1

跳转到page2

组件下的路由属性

- 每个组件都有两个属性\$router 和 \$route
- \$router存放一些方法（push replace等）
- \$route存放一些属性

路由的嵌套

```

<!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>Vue路由</title>
  <link rel="stylesheet" href="index.css">
  <!-- 0.引入 -->
  <script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
  <script src="https://cdn.bootcss.com/vue-router/3.0.6/vue-router.min.js"></script>
  <style>
    .active{
      color: red;
      background: #ccc;
    }
  </style>
</head>
<body>
  <div class="main" id="app1">
    <router-link to="/home">首页</router-link>
    <router-link to="/page2">页面2</router-link>
    <router-view></router-view>
  </div>
  <template id="home">
    <h1>首页</h1>
  </template>
  <!-- page2模版 -->
  <template id="page2">
    <div>
      <h1>page2</h1>
      <!-- 路由链接的to要写相对全路径 -->
      <router-link to="/page2/link1">link1</router-link>
      <router-link to="/page2/link2">link2</router-link>
      <router-view></router-view>
    </div>
  </template>
  <script>
    //
    var home = {template: "#home"};
    var page2 = {template: "#page2"};
    var link1 = {template: "<h2>link1</h2>"};
    var link2 = {template: "<h2>link2</h2>"};

    //3.routes映射直接写在router定义里
    var router = new VueRouter({
      routes: [
        {path: "/", component: home},
        {path: "/home", component: home},
        {
          path: "/page2",
          component: page2,

```

```

        children: [
            //子路由的路径不加斜杠
            {path: "", component: link1},
            {path: "link1", component: link1},
            {path: "link2", component: link2}
        ],
        {path: "/*", redirect: "/home"},
    ],
    linkActiveClass: "active",
  });

  new Vue({
    el: "#app1",
    data: {},
    router: router //注入 路由
  })
</script>
</body>
</html>

```

效果如下：



路由传参

```

<div class="main" id="app1">
  <router-link to="/list/1">list1</router-link><!-- 1.路径传参 -->
  <router-link to="/list/2">list2</router-link>
  <router-link to="/list/3">list3</router-link>
  <router-view></router-view>
</div>
<script>

var list = {template: "<h1>list{{$route.params.id}}</h1>"}; //3.输出id参数
var router = new VueRouter({
  routes: [
    {path: "/", component: list},
    {path: "/list/:id", component: list}, //2.传参
  ],

```

```

    linkActiveClass:"active",
  });

  new Vue({
    el:"#app1",
    data:{},
    router:router//注入 路由
  })
</script>

```

效果如下：



传递多个参数

```

<div class="main" id="app1">
  <router-link to="/list/1/list1">list1</router-link><!-- 1.路径传参 -->
  <router-link to="/list/2/list2">list2</router-link>
  <router-link to="/list/3/list3">list3</router-link>
  <router-view></router-view>
</div>
<script>

  var list = {template:"<h1>list{{$route.params.id}}, name:{{$route.params.name}}
  </h1>"}; //3.输出id参数
  var router = new VueRouter({
    routes:[
      {path:"/", component:list},
      {path:"/list/:id/:name", component:list}, //2.传多个参
    ],
    linkActiveClass:"active",
  });
</script>

```

效果如下：

file:///U:/codes/clcfnc/index.4.html#/list/1/list1

谷歌 网址大全 360搜索 游戏中心

[list1](#) [list2](#) [list3](#)

list1 , name:list1

参数监控与数据请求

```
var list = {
  template: "<h1>list{{$route.params.id}}, name:{{$route.params.name}}</h1>",
  watch: { // 监测
    $route() { // 只要$route下的属性发生变化
      // 做数据请求
    }
  }
};
```

一个Flex布局的SPA简单实例

```
<!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>Vue路由</title>
  <link rel="stylesheet" href="index.css">
  <!-- 0.5引入 -->
  <script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
  <script src="https://cdn.bootcss.com/vue-router/3.0.6/vue-router.min.js"></script>
  <style>
    .active{
      color: red;
      background: #ccc;
    }
    #app1{
      display: flex;
      flex-direction: row;
      flex-wrap: nowrap;
    }
    #left{
      width: 200px;
      background: #eee;
    }
    #left a{
      display: block;
      line-height: 30px;
```



```

padding: 2px;
}
#right{
width:500px;padding: 10px;
background: rgb(173, 14, 14);
}
</style>
</head>
<body>
  <div class="main" id="app1">
    <div id="left">
      <router-link :to="{name:'list',params:{id:1}}">list1</router-link>
      <router-link :to="{name:'list',params:{id:2}}">list2</router-link>
      <router-link :to="{name:'list',params:{id:3}}">list3</router-link>
    </div>
    <div id="right">
      <router-view></router-view>
    </div>

  </div>
<script>

var list ={
  template:"<h1>list{{$route.params.id}}</h1>",
  watch:{//监测
    $route(){//只要$route下的属性发生变化
      //做数据请求
    }
  }
};

var router =new VueRouter({
  routes:[
    {path:"/",component:list},
    {path:"/list/:id/:name",component:list,name:"list"},//设定路由的name
  ],
  linkActiveClass:"active",
});

new Vue({
  el:"#app1",
  data:{},
  router:router//注入 路由
})
</script>
</body>
</html>

```

效果如下：



采用数组v-for写法：

```
<!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>Vue路由</title>
  <link rel="stylesheet" href="index.css">
  <!-- 0.引入 -->
  <script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
  <script src="https://cdn.bootcss.com/vue-router/3.0.6/vue-router.min.js"></script>
  <style>
    .active{
      color: red;
      background: #ccc;
    }
    #app1{
      display: flex;
      flex-direction: row;
      flex-wrap: nowrap;
    }
    #left{
      width: 200px;
      background: #eee;
    }
    #left a{
      display: block;
      line-height: 30px;
      padding: 2px;
    }
    #right{
      width: 500px; padding: 10px;
      background: rgb(173, 14, 14);
    }
  </style>
</head>
```

```
<body>
  <div class="main" id="app1">
    <div id="left">
      <router-link :to="{name:'list',params:{id:item}}" v-for="item in
arr">link{{item}}</router-link>
    </div>
    <div id="right">
      <router-view></router-view>
    </div>

  </div>
</script>

var list = {
  template: "<h1>list{{$route.params.id}}</h1>",
  watch: { //监测
    $route() { //只要$route下的属性发生变化
      //做数据请求
    }
  }
};

var router = new VueRouter({
  routes: [
    {path: "/", component: list},
    {path: "/list/:id/:name", component: list, name: "list"}, //设定路由的name
  ],
  linkActiveClass: "active",
});

new Vue({
  el: "#app1",
  data: {
    arr: [1, 2, 3, 4, 5]
  },
  router: router //注入 路由
})
</script>
</body>
</html>
```