使用 vue-router2.0 创建单页导航

一、实验简介

1.1 实验内容

vue-router 是 Vue.js 官方的路由插件，适合用于构建单页面应用，本实验用来创建官方示例。

1.2 实验知识点

- vue-router.js 获取
- vue-router.js 如何使用
- router-link
- router-view
- transition

1.3 实验环境

- sublime
- ubuntu

1.4 适合人群

本课程难度为一般，属于初级级别课程，适合具有 html 和 js 经验的小伙伴。

1.5 代码获取

```
http://labfile.oss.aliyuncs.com/courses/762/vue-route.zip
```

本实验使用所有代码均已上传

二、课程介绍

vue-router 是 Vue.js 官方的路由插件，适合用于构建单页面应用。 vue 的单页面应用是基于路由和组件的，路由用于设定访问路径，并将路径和组件映射起来。在 vue-router 单页面应用中，组件的切换，使页面效果更加友好。我们单间的把所有的代码都写到一个文件里，这样比较直观的看到每一块代码的含义。

三、实验步骤

打开终端，进入 Code 目录，将其作为课程的工作目录。

启动 php 内置服务器

```
sudo php -S localhost:80
```

在实验环境中 80 端口并没有被占用，大家在使用中如果被占用的话，可以使用别的端口。例如使用8080端口的话，我们在终端执行下面的命令

```
sudo php -S localhost:8080
```

在浏览器访问时需在加上8080端口访问

```
localhost:8080
```

把代码下载到 Code 目录

下载文件到此目录

```
wget http://vuejs.org/js/vue.js
wget https://unpkg.com/vue-router@2.2.1/dist/vue-router.js
```

使用 vue-router 的基本步骤为

- 引入 vue.js 和 vue-router.js

```
<script src="./vue.js"></script>
<script src="./vue-router.js"></script>
```

- 创建组件

```
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!-- 使用 router-link 组件来导航. -->
    <!-- 通过传入 `to` 属性指定链接. -->
    <!-- <router-link> 默认会被渲染成一个 `<a>` 标签 -->
    <router-link to="/foo">Go to Foo</router-link>
    <router-link to="/bar">Go to Bar</router-link>
  </p>
  <!-- 路由出口 -->
  <!-- 路由匹配到的组件将渲染在这里 -->
  <router-view></router-view>
</div>
```

在上面代码中我们使用了 router-link 和.

- router-link 使用语法请参考 <a> 的用法

<router-link> 组件支持用户在具有路由功能的应用中（点击）导航。 通过 to 属性指定目标地址，默认渲染成带有正确链接的 <a> 标签，可以通过配置 tag 属性生成别的标签.。另外，当目标路由成功激活时，链接元素自动设置一个表示激活的 CSS 类名 router-link-active 。

```
<!-- 渲染成li标签 -->
<router-link to="home" tag="li"></router-link>
```

- router-view

<router-view> 组件是一个 functional 组件，渲染路径匹配到的视图组件。 <router-view> 渲染的组件还可以内嵌自己的 <router-view> ，根据嵌套路径，渲染嵌套组件。

- 创建路由

```
// 0. 如果使用模块化机制编程，导入Vue和VueRouter，要调用 Vue.use(VueRouter)

// 1. 定义（路由）组件。
// 可以从其他文件 import 进来
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

// 2. 定义路由
// 每个路由应该映射一个组件。 其中"component" 可以是
// 通过 Vue.extend() 创建的组件构造器，
// 或者，只是一个组件配置对象。
// 我们晚点再讨论嵌套路由。
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]

// 3. 创建 router 实例，然后传 `routes` 配置
// 你还可以传别的配置参数, 不过先这么简单着吧。
const router = new VueRouter({
  routes // （缩写）相当于 routes: routes
})

// 4. 创建和挂载根实例。
// 记得要通过 router 配置参数注入路由，
// 从而让整个应用都有路由功能
const app = new Vue({
  router
}).$mount('#app')

// 现在，应用已经启动了!
```

完整代码

创建 demo1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>实验楼--vue-route使用</title>
    <script src="./vue.js"></script>
    <script src="./vue-router.js"></script>
</head>
<body>
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!-- 使用 router-link 组件来导航. -->
    <!-- 通过传入 `to` 属性指定链接. -->
    <!-- <router-link> 默认会被渲染成一个 `<a>` 标签 -->
    <router-link to="/foo">Go to Foo</router-link>
    <router-link to="/bar">Go to Bar</router-link>
  </p>
  <!-- 路由出口 -->
  <!-- 路由匹配到的组件将渲染在这里 -->
  <router-view></router-view>
</div>
<script>
// 0. 如果使用模块化机制编程，导入Vue和VueRouter，要调用 Vue.use(VueRouter)

// 1. 定义（路由）组件。
// 可以从其他文件 import 进来
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

// 2. 定义路由
// 每个路由应该映射一个组件。 其中"component" 可以是
// 通过 Vue.extend() 创建的组件构造器，
// 或者，只是一个组件配置对象。
// 我们晚点再讨论嵌套路由。
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]

// 3. 创建 router 实例，然后传 `routes` 配置
// 你还可以传别的配置参数, 不过先这么简单着吧。
const router = new VueRouter({
  routes // （缩写）相当于 routes: routes
})

// 4. 创建和挂载根实例。
// 记得要通过 router 配置参数注入路由，
// 从而让整个应用都有路由功能
const app = new Vue({
  router
}).$mount('#app')

// 现在，应用已经启动了!
</script>
</body>
</html>
```

效果如下

上述官方示例简单描述了 vue-route 的用法，我们可以看到我们在点击两个链接的时候，页面并没有刷新，路由定义的地址在url上作出了体现。

下面我们做下一个实验示例。

示例中使用到 bootstrap ， 本文中不再详细解释

我们看到实验楼的导航列表中有几个菜单，我们点击不同菜单出现不同内容。好的现在实验开始。首先把创建一个实验楼的导航，下面是一个纯html的，我们接下来使用 v-router 来实现我们的需求。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>实验楼--vue-route使用</title>
    <link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    <script src="./vue.js"></script>
    <script src="./vue-router.js"></script>

</head>
<body>
 <nav class="navbar navbar-inverse">
     <div class="container">

       <div class="navbar-header">
          <img src="https://static.shiyanlou.com/img/logo_03.png" alt="" height="50">

       </div>
       <div id="navbar" class="collapse navbar-collapse">
         <ul class="nav navbar-nav">
           <li><a>课程</a></li>
           <li><a>路径</a></li>
           <li><a>训练营</a></li>
           <li><a>讨论区</a></li>
         </ul>
       </div>
</nav>
    <div class="container">

       <h1>主页</h1>
    </div>

</body>
</html>
```

我们要实现的效果是我们点每一个菜单那么下面就切换成每个菜单对应的内容。

我们分析这个页面有几个路径.

路径	导航
/home	首页
/courses	课程

动手实践是学习 IT 技术最有效的方式!　　开始实验

路径	导航
/paths	路径
/bootcamp	训练营

每个路径我们对应一个组件

路径	导航	组件
/home	首页	Home
/courses	课程	Courses
/paths	路径	Paths
/bootcamp	训练营	Bootcamp

我们在代码中已经引入了 vue 和 vue-router 文件，

第一步创建组件

创建这四个组件

```
const Home = { template: '<div><h1>首页</h1></div>' }
const Courses = { template: '<div><h1>课程</h1></div>' }
const Paths = { template: '<div><h1>路径</h1></div>' }
const Bootcamp = { template: '<div><h1>训练营</h1></div>' }
```

组件创建完毕

创建路由

```
var router = new VueRouter()
```

路在被定义为一个在 router 实例里的一个 routes 选项数组，定义路由，

然后把组件映射到路由

```
<!--1.x的映射方法-->
<!--router.map({-->
<!--    '/home': { component: Home },-->
<!--    '/courses': { component: Courses },-->
<!--    '/paths': { component: Paths },-->
<!--    '/bootcamp': { component: Bootcamp }-->
<!--})-->
<!--2.x的映射方法-->
var router = new VueRouter({
  routes: [
    { path: '/home', component: Home },
    { path: '/courses', component: Courses },
    { path: '/paths', component: Paths },
    { path: '/bootcamp', component: Bootcamp }
  ]
})
```

使用路由连接 router-link

```
<router-link to="home">Home</router-link>
<router-link to="courses">Courses</router-link>
<router-link to="paths">Paths</router-link>
<router-link to="bootcamp">Bootcamp</router-link>
```

使用 <router-view> 来渲染组件

```
<!--在页面上写上<router-view></router-view>用来渲染组件-->
<router-view></router-view>
```

最后启动路由

```
const app = new Vue({
  router
}).$mount('#app')
```

完整代码

demo2.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>实验楼--vue-route使用</title>
    <link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    <script src="./vue.js"></script>
    <script src="./vue-router.js"></script>

</head>
<body>
<div id="app">
 <nav class="navbar navbar-inverse">
     <div class="container">

       <div class="navbar-header">
           <img src="https://static.shiyanlou.com/img/logo_03.png" alt="" height="50">
       </div>
       <div id="navbar" class="collapse navbar-collapse">
         <ul class="nav navbar-nav">
           <li><router-link to="home">首页</router-link></li>
           <li><router-link to="courses">课程</router-link></li>
           <li><router-link to="paths">路径</router-link></li>
           <li><router-link to="bootcamp">训练营</router-link></li>
         </ul>
       </div>
       </div>
</nav>
<div class="container">
<router-view></router-view>
</div>
<script>
const Home = { template: '<div><h1>首页</h1></div>' }
const Courses = { template: '<div><h1>课程</h1></div>' }
const Paths = { template: '<div><h1>路径</h1></div>' }
const Bootcamp = { template: '<div><h1>训练营</h1></div>' }

var router = new VueRouter({
  routes: [
    { path: '/home', component: Home },
    { path: '/courses', component: Courses },
    { path: '/paths', component: Paths },
    { path: '/bootcamp', component: Bootcamp }
  ]
})

const app = new Vue({
  router
}).$mount('#app')
</script>
</body>
</html>
```

效果如下

此次示例的流程

`vue 流程`

- 创建组件
- 创建路由
- 映射路由

`html`

- `router-link 渲染 router-view`

现在我们实现了第一步，下一步要给选中的菜单加上单独的样式。

现在我们要用到 `router-link` 的一个属性，在一个 `router-link` 被选中的时候会增加一个**class**属性值 `router-link-active` 我们在样式中定义这个样式即可。我们在 `router-link` 被选中的时候字体变成白色。

```
.router-link-active{
       background-color: #ffffff ;
       color: #ff0000 ;
       border-radius:5px;
   }
```

好我们在页面中加入样式后，发现在使用**bootstrap**后我们的自定义样式不起作用，我们需要给我们自己的样式添加权重

```
.router-link-active{
      background-color: #ffffff !important;
       color: #ff0000 !important;
       border-radius:5px;
   }
```

添加样式保存为 `demo3.html`

效果图如下

现在把选中样式添加以后，在给 router-view 中的切换带动画效果。

<router-view> 是基本的动态组件，所以我们可以用 <transition> 组件给它添加一些过渡效果：

transition 默认有一个效果。我们也可以直接给transition添加一些动态效果。

- transition
  - v-enter：定义进入过渡的开始状态。在元素被插入时生效，在下一个帧移除。
  - v-enter-active：定义进入过渡的结束状态。在元素被插入时生效，在 transition/animation 完成之后移除。
  - v-leave：定义离开过渡的开始状态。在离开过渡被触发时生效，在下一个帧移除。
  - v-leave-active：定义离开过渡的结束状态。在离开过渡被触发时生效，在 transition/animation 完成之后移除。

渲染 router-view

```
<transition name="move">
    <router-view></router-view>
</transition>
```

我们定义两个简单的效果

```
.move-enter-active, .move-leave-active {
    transition: opacity .5s
}
.move-enter, .move-leave-active {
     opacity: 0.1
}
```

添加动画效果后保存为 demo4.html

在过渡结束和离开时我们把div的透明度设置成0.

效果如下

简单的设置一个效果以后呢，实验楼的课程下有三个选项，那这三个选项怎么在课程下体现呢，使用嵌套路由就可以实现这个效果。

接下来我们在课程的组件内添加一个新的路由

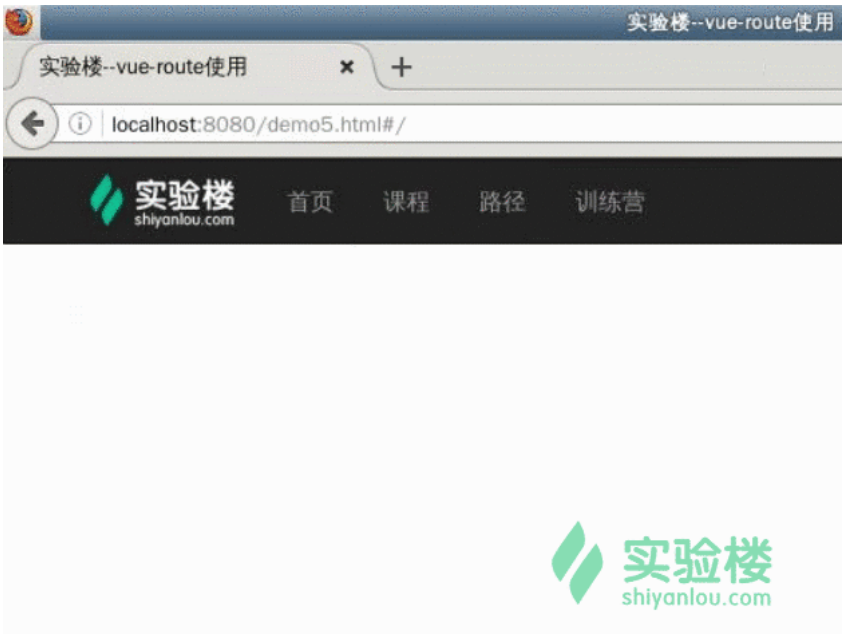
```
const Courses = { template: '<div><h1>课程</h1><ul><li><router-link to="/courses/all">全部课程</router-link></li><li><router-link to="/courses/priview">即将上线</router-link></li><li><router-link to="/courses/develop">开发者</router-link></li></ul><div><router-view></router-view></div></div>' }
```

然后在路由映射中添加子路由的映射，修改 courses 的路由

```
{ path: '/courses', component: Courses ,
      children:[
                {path:'all', component:coursesall},
                {path:'priview', component:coursespriview},
                {path:'develop', component:coursesdevelop}
            ]
 },
```

把代码保存为 demo5.html

现在效果图如下



我们修改下样式，让页面看的稍微好看点。

使用 bootstrap 给 courses 子路由中的 ul 添加 tab 切换样式

```
class="nav nav-tabs"
```

保存为 demo6.html

效果如下：

本次实验就先到这里。

四、实验总结

通过本次实验，大家可以简单的了解了 vue 路由的使用方法，可以知道如何获取和引入 vue-route ，创建简单的单应用导航页面，希望可以帮到各位小伙伴。

五、参考链接

参考链接 vue-route2.0官方文档 (http://router.vuejs.org/zh-cn/)

实验楼感谢您对我们的信任与支持！

下一节 > (/courses/762/labs/2576/document)

课程教师

Adward 工程师
某信息安全公司
共发布过**8**门课程

查看老师的所有课程 > (/teacher/79217)