

第十一章 vue-router路由和前端状态管理

11.1 vue-router路由基本加载

简单四步走

1. 安装

```
npm install --save vue-router
```

2. 引用

```
import router from 'vue-router'  
Vue.use(router)
```

3. 配置路由文件，并在vue实例中注入

```
var rt = new router({  
  routes:[{  
    path: '/',//指定要跳转的路径  
    component:HelloWorld//指定要跳转的组件  
  }]  
})  
new Vue({  
  el: '#app',  
  router:router,  
  components: { App },  
  template: '<App/>'  
})
```

4. 确定视图加载的位置

```
<router-view></router-view>
```

11.2 vue-router路由的跳转

```

<router-link to="/"></router-link>

<template>
  <ul>
    <li>
      <router-link to="/helloworld">HELLO WORLD</router-link>
    </li>
    <li>
      <router-link to="/helloearth">HELLO EARTH</router-link>
    </li>
  </ul>
</template>

```

11.3 vue-router路由参数的传递

1. 必须在路由内加入路由的name
2. 必须在path后加/: +传递的参数

1. 传递参数和接收参数看下边代码

```

<router-link
  :to="{name: helloearth,params:{msg: 只有一个地球}}">
  HELLO WORLD
</router-link>

```

读取参数: \$route.params.XXX
 方式: ===/helloworld/你好世界

```

<router-link
  :to="{path: '/helloearth',query:{msg: 只有一个地球}}">
  HELLO WORLD
</router-link>

```

方式: ===/helloworld?name=XX&count=xxx
 函数模式

你可以创建一个函数返回 props。这样你便可以将参数转换成另一种类型，将静态值与基于路由的值结合等等。

```

const router = new VueRouter({
  routes: [
    { path: '/search', component: SearchUser, props: (route) => ({

```

```
query: route.query.q }) }  
  ]  
})
```

11.3.1 Axios之get请求详解

axios的简介：

axios 是一个基于Promise 用于浏览器和 nodejs 的 HTTP 客户端，它本身具有以下特征：

- 从浏览器中创建 XMLHttpRequest
- 从 node.js 发出 http 请求
- 支持 Promise API
- 拦截请求和响应
- 转换请求和响应数据
- 取消请求
- 自动转换JSON数据
- 客户端支持防止 CSRF/XSRF

1. 安装

```
npm install axios
```

2. 引入加载

```
import axios from 'axios'
```

3. 将axios全局挂载到VUE原型上

```
Vue.prototype.$http = axios;
```

4. 发出请求 以cnode社区API为例子

```
// 为给定 ID 的 user 创建请求  
使用传统的function  
getData(){  
  var self = this;  
  this.$http.get('https://cnodejs.org/api/v1/topics')
```

```

        .then(function (res) {
            //此处的this指向的不是当前vue实例
            self.items = res.data.data
            console.log(res.data.data)
        })
        .catch(function (err) {
            console.log(err)
        })
    }

    // 可选地，上面的请求可以这样做
    两种传递参数的形式
    axios.get('/user', {
        params: {
            ID: 12345
        }
    })
    axios.get('/user', {
        ID: 12345
    })

    -----
    axios.get('https://cnodejs.org/api/v1/topics?page=1&limit=15')

```

使用CNODE社区官方的API为例展开学习

获取主题列表API : <https://cnodejs.org/api/v1/topics>

参数 : page页码

limit 每页显示的数量

11.3.1 Axios之post请求详解

```

// 为给定 ID 的 user 创建请求
使用传统的function
getData(){
    var self = this;
    this.$http.post(url,{
        page:1,
        limit:10
    })
    .then(function (res) {
        //此处的this指向的不是当前vue实例
        self.items = res.data.data
        console.log(res.data.data)
    })
    .catch(function (err) {

```

```
        console.log(err)
    })
}
```

POST传递数据有两种格式：

- form-data ?page=1&limit=48
- x-www-form-urlencoded { page: 1, limit: 10 }

在axios中，post请求接收的参数必须是form-data
qs插件——qs.stringify

11.4 Vuex之store

用来管理状态，共享数据，在各个组件之间管理外部状态
如何使用？

第一步：引入vuex，并通过use方法使用它

第二步：创建状态仓库

第三步：通过this.\$store.state.XXX直接拿到需要的数据

```
//创建状态仓库，注意Store, state不能改
var store = new Vuex.Store({
  state:{
    XXX: xxx
  }
})
//直接通过this.$store.state.XXX拿到全局状态
```

11.5 Vuex的相关操作

vuex状态管理的流程

view——>actions——>mutations——>state——>view

除了能够获取状态如何改变状态呢？

```
//创建状态仓库，注意Store, state不能改
var store = new Vuex.Store({
  state:{
    XXX: xxx
  },
  mutations:{
```

```
}  
})
```

```
this.$store.commit(XXX);
```

此处的XXX是你在mutations中定义的方法名

```
var store = new Vuex.Store({  
  state:{  
    XXX: xxx  
  },  
  mutations:{  
    a:function(state){  
    }  
  },  
  actions:{  
    b:function(context){  
      context.commit('a');  
    }  
  }  
})
```

如何调用

```
this.$store.dispatch(XXX);
```

```
getters:{  
  
}  
this.$store.getters.getCount
```

注意：actions提交的是mutation,而不是直接变更状态
actions可以包含异步操作，但是mutation只能包含同步操作