**Vue第四天**

## Vue前端助手 **下载，网盘地址：**

[vue-devtools-4.1.4\_0下载](https://pan.baidu.com/s/1O41BdUzTdmkj8AX4uhwcVA)  
密码：713

**一、axios和vue-axios的使用**

区别：

1. axios是一个库，并不是vue中的第三方插件，使用时不能通过Vue.use()安装插件，需要在原型上进行绑定
2. vue-axios是将axios集成到Vuejs的小包装器，可以像插件一样进行安装

##推荐使用vue-axios，因为直接安装的方式更符合Vue的使用规范

先安装：cnpm install --save axios vue-axios

import Axios from 'axios'

import VueAxios from 'vue-axios'

Vue.prototype.$http = Axios

Vue.use(VueAxios, Axios)

使用方法：

this.$http.get('/api/w/website/bannerList').then(res=>{

console.log(res);

})

1. **跨域**

定义：指的是浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器施加的安全限制。

同源：域名，协议，端口均相同，不明白没关系

www：（World Wide Web） 万维网

<https://blog.csdn.net/>（博客网址）

协议：//子域名.主域名（可能会有端口号）

网址的构成：http协议：//www(主机名).域名/文件名

域名包括主域名和子域名

* http://www.123.com/index.html 调用 http://www.123.com/server.php （非跨域）
* http://www.123.com/index.html 用 http://www.456.com/server.php （主域名不同:123/456，跨域）
* http://abc.123.com/index.html 调用 http://def.123.com/server.php （子域名不同:abc/def，跨域）
* http://www.123.com:8080/index.html 调用 http://www.123.com:8081/server.php （端口不同:8080/8081，跨域）
* http://www.123.com/index.html 调用 https://www.123.com/server.php （协议不同:http/https，跨域）

Vue文件请求接口

this.$http.get('https://www.wumeili.to/w/website/bannerList').then(res=>{

console.log(res);

})

刷新页面，控制台报错如下

注：当以后再控制台出现XHMHttpRequest和Access-Control-Allow-Origin

都是跨域



**跨域的解决方法：（vue中用反向代理解决跨域）**

在config/index.js文件中配置（配置好了，记得重启项目哦）

proxyTable: {

'/api': {

target: 'https://www.wumeili.top',// 接口域名

// secure: false, // 如果是https接口，需要配置这个参数

changeOrigin: true,//是否跨域

pathRewrite: { //需要rewrite的

'^/api': 'https://www.wumeili.top'

}

},

},

然后在页面中使用

this.$http.get('/api/w/website/bannerList').then(res=>{

console.log(res);

})

1. **vuex**

访问的网址：<https://vuex.vuejs.org/zh/>

Vuex 是一个专为 Vue.js 应用程序开发的****状态管理模式****。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化

这个状态自管理应用包含以下几个部分：

* ****state****，驱动应用的数据源；
* ****view****，以声明方式将 ****state**** 映射到视图；
* ****actions****，响应在 ****view**** 上的用户输入导致的状态变化。

每一个 Vuex 应用的核心就是 store（仓库）。“store”基本上就是一个容器，它包含着你的应用中大部分的****状态 (state)****。Vuex 和单纯的全局对象有以下两点不同：

* Vuex 的状态存储是响应式的。
* 你不能直接改变 store 中的状态。

**在src里面新建一个文件夹store，然后在store下面创建一个index.js**在index.js写一个初始代码

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

const store = new Vuex.Store({

state: {},

})

export default store //需要导出

需要在main.js中引入一下（类似于router）：

import store from './store'

new Vue({

el: '#app',

router,

store, //必须加上

components: { App },

template: '<App/>'

})

**案例1：（简单的运算）还与store没有关系**

**Template:**

<template>

<div>

<h2>简单的计算器</h2>

<div>

<input type="button" value="-" @click="jian"/>

<span>{{num}}</span>

<input type="button" value="+" @click="add">

</div>

</div>

</template>

1. **Script**

export default {

data() {

return {

num:100,

}

},

methods:{

add(){ this.num+=5; },

jian(){ this.num-=5;}

}

}

**案例2（运算使用vuex）**

1. **Template（没变）**
2. **Script**

export default {

data() {

return {

// 在vue助手中可以看到数据的变化，但是在页面上面没有改变

num:this.$store.state.num

}

},

methods:{

add(){

this.num+=5;

},

jian(){

this.num-=5;

}

}

}

</script>

**注：我们可以看到前端助手上面的数据没有发生改变，页面发生了改变**

**解决此问题：**

**改变store/ index.js文件**

const store = new Vuex.Store({

state: {

num: 105

},

mutations: {

increment(state) {

// 函数接收一个参数state,这个参数就是上面的state

state.num += 5;

},

mincrement(state) {

state.num -= 3;

}

**在组件中的script修改为 this.$store.commit('increment')**

add(){

// this.num+=5;

// 通过这提交一个mutations，commit里面提交的就是mutations的函数名字

this.$store.commit('increment');

},

**通过以上操作就可以在vue助手中看到数据同步，但是页面的数据没有发生改变，怎么办呢？请看案例3**

**案例3 （computed数据同步）**

**说明：**computed用来监控自己定义的变量，该变量不在data里面声明，直接在computed里面定义，然后就可以在页面上进行双向数据绑定展示出结果或者用作其他处理；

**与watch的区别：**

watch主要用于监控vue实例的变化，watch监控的变量当然必须在data里面声明才可以，watch可以监控一个变量，watch可以是一个对象

1. **Template（不变）**
2. **Script**

export default {

// data() {

// return {

// num:this.$store.state.num

// }

// },

computed:{

// 在vue助手中可以看到数据的变化，同事在页面上面数据也改变

num(){

return this.$store.state.num

}

},

**现在你就可以看到vue助手的内容和页面的内容是同步变化的**

**案例4 （vuex的传参：1个参数）**

**1、Template**

<template>

<div>

<h2>简单的计算器</h2>

<div>

<input type="button" value="-" @click="jian(3)"/>

<span>{{num}}</span>

<input type="button" value="+" @click="add(2)">

</div>

</div>

</template>

**2、Script**

methods:{

add(num){

this.$store.commit('increment',num);

},

jian(num){

// this.num-=5;

this.$store.commit('mincrement',num);

}

}

1. **store/index.js**

mutations: {

// 组件中函数传一个参数

increment(state, pload) {

// 函数接收一个参数state,这个参数就是上面的state

state.num += pload;

},

mincrement(state, pload) {

state.num -= pload;

}

}

**案例5 （vuex的传参：多个参数）**

1. **Template（不变）**
2. **Script**

methods:{

add(num){

// this.num+=5;

// 通过这提交一个mutations，commit里面提交的就是mutations的函数名字

//方法一

this.$store.commit('increment',{n:num,name:'zhangsan'});

//方法二

// type类型

this.$store.commit({

type:'increment',

n:num,

name:'wangxiaoer'

});

},

jian(num){

// this.num-=5;

this.$store.commit('mincrement',{n:num,name:'zhangsan'});

}

}

1. **store/index.js**

mutations: {

// 组件中函数传多个个参数

increment(state, pload) {

// 函数接收一个参数state,这个参数就是上面的state

console.log(pload);

state.num += pload.n;

},

mincrement(state, pload) {

state.num -= pload.n;

}

}

**案例5 （vuex的异步请求）**

**当异步请求数据时**

**1、Template（不变）**

**2、Script（不变）**

**3、store/index.js**

mutations: {

// 异步请求时，

increment(state, pload) {

// 如果在这里面写异步的话，状态会发生混乱

// mutations的设计是同步，解决办法使用action

setTimeout(() => {

state.num += pload.n;

}, 1000);

// state.num += pload.n;

},

mincrement(state, pload) {

state.num -= pload;

}

}

**当你写了setTImeout（延时器）时，你会发现vue助手中的数据和页面的数据发生了混乱**

**原因：因为mutations的设计是同步，而我们触发的是异步，**

**解决办法：使用action请看案例6;**

**案例6 （vuex的action使用方法）**

**Action 类似于 mutation，不同在于：**

* Action 提交的是 mutation，而不是直接变更状态。
* Action可包含任意的异步操作

**1、Template（不变）**

**2、Script**

methods:{

add(num){

// Action 通过 store.dispatch 方法触发：

this.$store.dispatch('Addaction');

},

jian(num){

// this.num-=5;

this.$store.commit('mincrement',{n:num,name:'zhangsan'});

}

}

**3、store/index.js**

mutations: {

increment(state, pload) {

state.num += pload.n;

},

mincrement(state, pload) {

state.num -= pload;

}

},

actions: {

Addaction(context) {

console.log(context);

setTimeout(() => {

context.commit('increment', { n: 5 });

}, 1000);

context.dispatch("textAction");

},

// action之间相互调用

textAction() {

console.log("context触发了")

}

}

**注:进行以上操作后，打开vue项目观察页面数据和vue助手中数据的变化，现在可以保持同步变化**

**案例7 （vuex的getters使用方法）**

**store 中的 state 中派生出一些状态，例如对列表进行过滤并计数：**

**getters就是复制了一份state里面的状态，在这个里面操作后返回，类似于计算属性**

1. **Template**

<template>

<div>

<h2>简单的计算器</h2>

<div>

<input type="button" value="-" @click="jian(3)"/>

<span>{{num}}</span>

<input type="button" value="+" @click="add(2)">

{{num2}}

</div>

</div>

</template>

1. **Script**

export default {

// data() {

// return {

// num:this.$store.state.num

// }

// },

computed:{

// 在vue助手中可以看到数据的变化，同事在页面上面数据也改变

num(){

return this.$store.state.num

},

num2(){

// return this.num > 120 ? 120 : this.num;

return this.$store.getters.count;

}

},

methods:{

add(num){

// Action 通过 store.dispatch 方法触发：

this.$store.dispatch('Addaction');

},

jian(num){

// this.num-=5;

this.$store.commit('mincrement',{n:num,name:'zhangsan'});

}

}

}

1. **store/index.js**

const store = new Vuex.Store({

state: {

num: 105

},

getters: {

count: state => {

return state.num > 120 ? 120 : state.num

}

},

})

export default store

**案例8 （vuex的辅助函数使用方法）**

vuex提供了辅助函数处理庞大的vuex数据，mapState，mapGetters，mapMutations，mapActions，实际就是把state、getters、mutations、actions整合成一个数组，一次性返回

1. **Template（不变）**

<template>

<div>

<h2>简单的计算器</h2>

<div>

<input type="button" value="-" @click="jian(3)"/>

<span>{{num}}</span>

<input type="button" value="+" @click="add(2)">

<h3> <font color="red">{{name}}</font> </h3>

<h3> <font color="blue">{{age}}</font> </h3>

<h1> {{count}} </h1>

</div>

</div>

</template>

1. **Script**

import { mapGetters,mapState,mapActions } from 'vuex'

export default {

computed:{

...mapState({

num(){

return this.$store.state.num

},

name(){

return this.$store.state.name

},

age(){

return this.$store.state.age

},

}),

...mapGetters({

count:'count'

})

},

methods:{

// mapActions要在methods中引入

...mapActions({

addhandle:'Addaction'

}),

add(num){

// Action 通过 store.dispatch 方法触发：

this.$store.dispatch('Addaction');

},

jian(num){

// this.num-=5;

this.$store.commit('mincrement',{n:num,name:'zhangsan'});

}

}

1. **store/index.js（不变）**

##### **注：**

##### **如果要引入mapMutations辅助函数，那么组件中函数需要this.$store.commit(‘函数名’)来提交函数**

* **如果要引入mapActions辅助函数，那么组件中函数需要this.$store.dispatch(‘actions的函数名’)来出发函数函数**