

vuex全局状态管理

一、什么是全局状态

要弄清楚这个问题之前，先要搞清楚一个点就是为什么需要全局状态？

在之前学习vue的时候，我们都知道如果想实现组件之间的传值，我们有以下几种方式

1. 父级组件向子级组件传值使用 `props`
2. 子级组件向父级组件传值使用，插槽，`emit` 自定义事件
3. 跨级组件传值使用 `provide/inject` 【vue3版本支持,这一种方案实现的全局数据因为要遵守数据流的单向性，所以子级组件不能改变全局注入进来的值】

第一个组件

第二个组件

`nickName = "张三"`

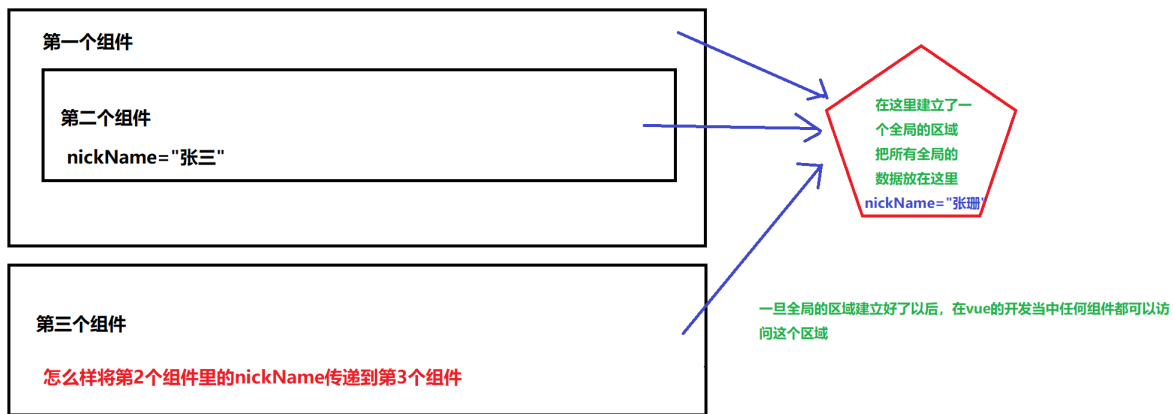
第三个组件

怎么样将第2个组件里的nickName传递到第3个组件

之前所讲过的三种数据传递方式，各有各的特点，`props` 适用于简单的父子组件传递，`emit` 以及也只适用于父子级件传递，而 `provide/inject` 虽然可以实现跨级的传递，但是并不能实现这种兄弟或跨了很多组件的兄弟传递

对于这一种复杂的数据共享问题，vue的内部也做了相应的技术解决方法，目前提出的最好的就说过就是全局状态管理

全局状态管理可以认为全局数据管理



二、vue的全局状态管理工具

根据上面的需求，我们知道现在需要一个全局的区域去存储我们的数据，这个区域怎么建立，怎么操作呢？在vue的全家桶下面，要实现这个方案有很多种办法，目前比较流行的有2种

1. `vuex` 全局状态管理
2. `pinia` 全局状态管理【vue3的composition api里面讲解】

三、vuex的安装

```
1 $ npm install vuex --save;
```

如果使用的是 `script` 标签引入的，可以直接导入

```
1 <script src="./js/vuex.global.js"></script>
```

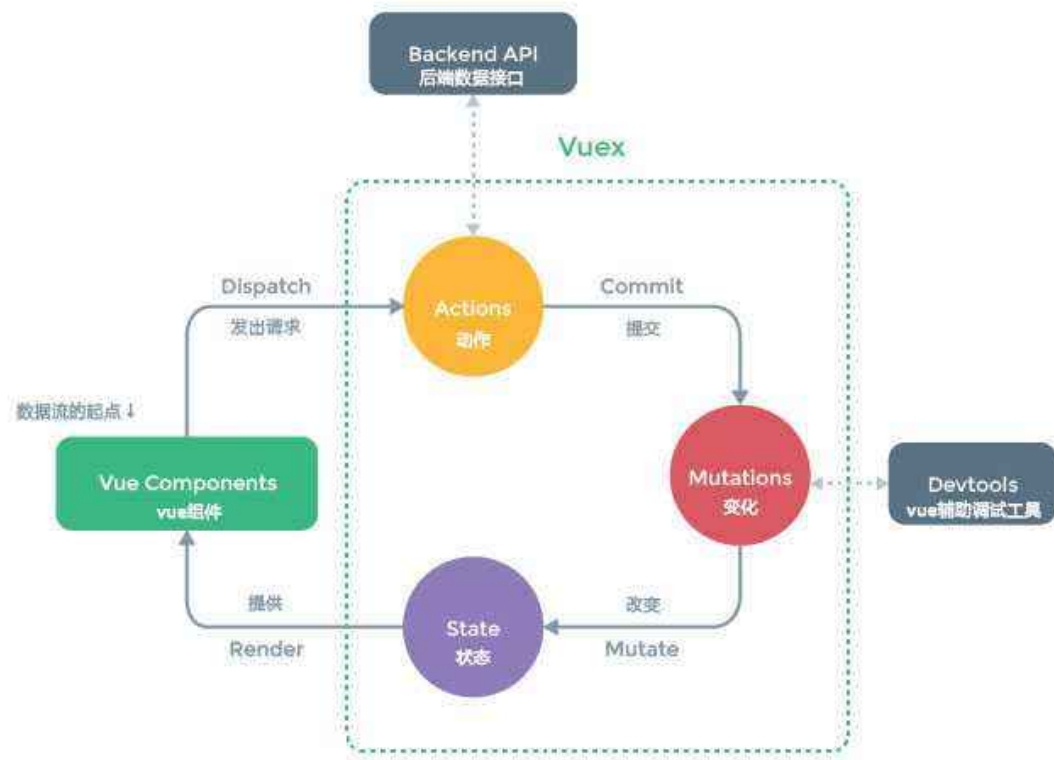
创建store

```
1 //第一步：先创建了vuex的store
2 const store = Vuex.createStore({
3
4 });
```

加载store

```
1 app.use(store);
```

四、vuex内部原理



1.state状态

可以认为 `state` 就是vue里面的 `data`，它用于存储全局的状态，也就是全局的变量

```
1  state: {
2    nickName: "张珊"
3  }
```

当我们在 `state` 下面定义了一个变量以后，这样在所有的组件里面就可以通过 `this.$store.state.nickName` 去拿到 `nickName` 的值

这里要注意，如果要改变state的值就不能直接改变

组件只能使用state里面的值，不能改变state里面的值。如果要改变，必须遵守 `vuex` 内部的流程

2. mutations转变

这个东西就是用于改变 `state` 里面的数据的

```
1  //它是用于改变state的，所有的状态改变应该都是通过`mutations`，它就相当于锦衣卫，
2  //里面所有的东西都是一个方法，系统会在所有方法的第一个参数上面注入状态state
3  mutations:{
4    setNickName(state,nickName){
5      state.nickName = nickName;
6    }
7  }
```

3.actions方法

```
1 //actions是用于提交任务的，相当于太监拿着皇帝的命令给了锦衣卫
2 //actions里面也全部都是一个方法，它会向所有的方法里注入第一个参数，这个参数是context
3 //我们可以从这个context里面解构出一上方法叫`comment`
4 actions:{
5     setNickName({commit},nickName){
6         commit("setNickName",nickName)
7     }
8 }
```

当我们完成上面的东西以后，我们就可以通过下面的代码来完成全局状态的改变

```
1 this.$store.dispatch("setNickName","樱木花道");
```

4.getters获取

在vuex的内部一个东西与我们之前在vue里面所学过的计算属性 `computed` 非常像，它叫 `getters`

```
1 getters:{
2     money(){
3         return ~~(Math.random()*1000);
4     }
5 }
```

5.vuex的总结

```
1 //第一步：先创建了vuex的store
2 const store = Vuex.createStore({
3     // 相当于vue里面的data，可以存放数据，这里存放的数据可以在任何组件里面使用
4     state: {
5         nickName: "张珊",
6         stuList:[
7             {stuName:"张三",money:1000},
8             {stuName:"李四",money:2000}
9         ]
10    },
11    //它是用于改变state的，所有的状态改变应该都是通过`mutations`，它就相当于锦衣卫，
12    //里面所有的东西都是一个方法，系统会在所有方法的第一个参数上面注入状态state
13    mutations:{
14        setNickName(state,nickName){
15            console.log("我是mutations,我触发了");
16            state.nickName = nickName;
17        }
18    },
19    //actions是用于提交任务的，相当于太监拿着皇帝的命令给了锦衣卫
20    //actions里面也全部都是一个方法，它会向所有的方法里注入第一个参数，这个参数是context
21    //我们可以从这个context里面解构出一上方法叫`comment`
22    actions:{
23        setNickName({commit},nickName){
24            console.log("我是action,我触发了");
25            commit("setNickName",nickName)
26        }
27    },
28    //这个东西相当于vue里面的计算属性 computed
29    //在所有getters的方法里面，系统会注入第一个参数state
```

```

30     getters:{
31         money(state){
32             return ~~(Math.random()*1000);
33         },
34         totalMoney(state){
35             let sum = 0;
36             state.stuList.forEach(item=>{
37                 sum+=item.money;
38             });
39             return sum;
40         }
41     }
42 });

```

五、vuex的快速操作

在vuex里面，大家都知道，如果要获取一个状态的数据要通过 `$store.state.变量` 来完成，如果要操作一个方法，则需要使用 `$store.dispatch("方法名", "参数")` 来进行。但是这种方式操作起来太麻烦了

vuex提供了快速操作的方法

1. mapState

用于将state的值快速的绑定到 `computed`

对象语法

```

1  app.component("one", {
2      template: "#temp1",
3      computed:{
4          ...mapState({
5              nickName:state=>state.nickName
6          })
7      }
8  });

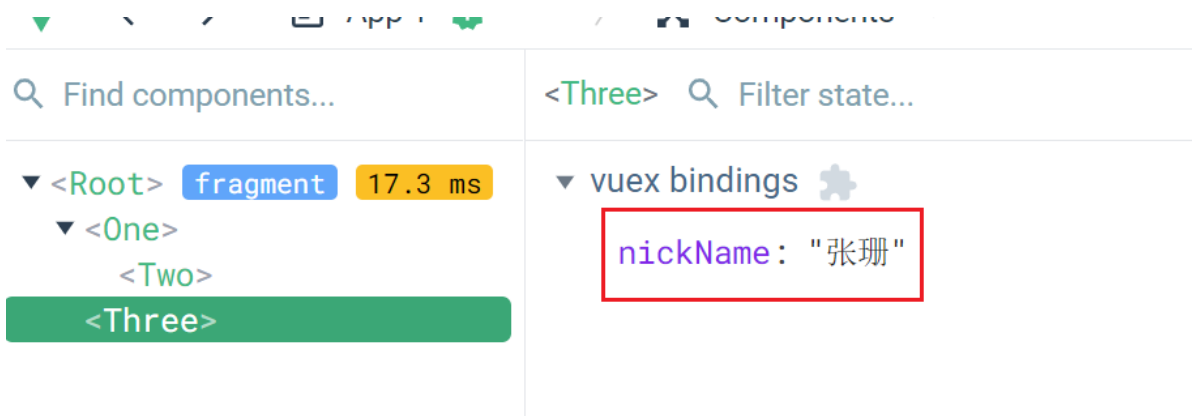
```

数组语法

```

1  app.component("three", {
2      template: "#temp3",
3      computed:{
4          ...mapState(["nickName"])
5      }
6  });

```



这个时候就可以直接使用 `nickName` 了

```
1 <h3>昵称: {{nickName}}</h3>
```

2.mapGetters

主要的作用就是快速的把 `getters` 里面的值绑定到 `computed`

```
1 computed:{
2   ...mapState({
3     nickName:state=>state.nickName
4   }),
5   ...mapGetters(["money", "totalMoney"])
6 }
```

这个地方使用的是数组语法，暂时发现不能使用对象语法

在上面的代码里面，我们就成功的快速拿到了 `money` 与 `totalMoney`

3.mapActions

主要的作用就是把 `vuex` 里在的 `actions` 映射到 `methods` 里面直接使用

```
1 methods:{
2   ...mapActions(["setNickName"]),
3   changeNickName(){
4     // this.$store.dispatch("setNickName", "樱木花道");
5     this.setNickName("流川枫");
6   }
7 }
```

最终代码

```
1 <!DOCTYPE html>
2 <html lang="zh">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>vuex的介绍</title>
9   <style>
10     .box {
```

```

11         border: 2px solid black;
12         padding: 4px;
13         margin: 4px;
14     }
15     </style>
16 </head>
17
18 <body>
19     <div id="app">
20         <one>
21             <two></two>
22         </one>
23         <three></three>
24     </div>
25     <template id="temp1">
26         <div class="box">
27             <h2>第1个组件</h2>
28             <h3>昵称: {{ $store.state.nickName }}</h3>
29             <h3>金额: {{ $store.getters.money }}</h3>
30             <h3>金额: {{ money }}</h3>
31             <h3>总金额: {{ totalMoney }}</h3>
32             <h3>昵称: {{ nickName }}</h3>
33             <slot></slot>
34         </div>
35     </template>
36     <template id="temp2">
37         <div class="box">
38             <h2>第2个组件</h2>
39             <button type="button" @click="changeNickName">我要改变
nickName</button>
40         </div>
41     </template>
42     <template id="temp3">
43         <div class="box">
44             <h2>第3个组件</h2>
45             <h3>昵称: {{ nickName }}</h3>
46         </div>
47     </template>
48 </body>
49 <script src="./js/vue.global.js"></script>
50 <script src="./js/vuex.global.js"></script>
51 <script>
52     // 如果想实现vuex的快速操作，一定要借用于它内部的工具
53     const {mapState,mapGetters,mapActions} = Vuex;
54     const app = Vue.createApp({
55         created() {
56             console.log(this.$store.state.nickName);
57         }
58     });
59
60     const store = Vuex.createStore({
61         state: {
62             nickName: "张珊",
63             stuList:[
64                 {stuName:"张三",money:1000},
65                 {stuName:"李四",money:2000}

```

```

66     ]
67   },
68   mutations:{
69     setNickName(state,nickName){
70       state.nickName = nickName;
71     }
72   },
73   actions:{
74     setNickName({commit},nickName){
75       commit("setNickName",nickName)
76     }
77   },
78   getters:{
79     money(state){
80       return ~~(Math.random()*1000);
81     },
82     totalMoney(state){
83       let sum = 0;
84       state.stuList.forEach(item=>{
85         sum+=item.money;
86       });
87       return sum;
88     }
89   }
90 });
91
92 app.use(store);
93
94 app.component("one", {
95   template: "#temp1",
96   computed:{
97     ...mapState({
98       nickName:state=>state.nickName
99     }),
100     ...mapGetters(["money","totalMoney"])
101   }
102 });
103 app.component("two", {
104   template: "#temp2",
105   methods:{
106     ...mapActions(["setNickName"]),
107     changeNickName(){
108       // this.$store.dispatch("setNickName","樱木花道");
109       this.setNickName("流川枫");
110     }
111   }
112 });
113 app.component("three", {
114   template: "#temp3",
115   computed:{
116     ...mapState(["nickName"])
117   }
118 });
119
120 app.mount("#app");
121 </script>

```



```
122  
123   </html>
```

六、vuex状态持久化

```
1  npm install vuex-persistedstate --save
```