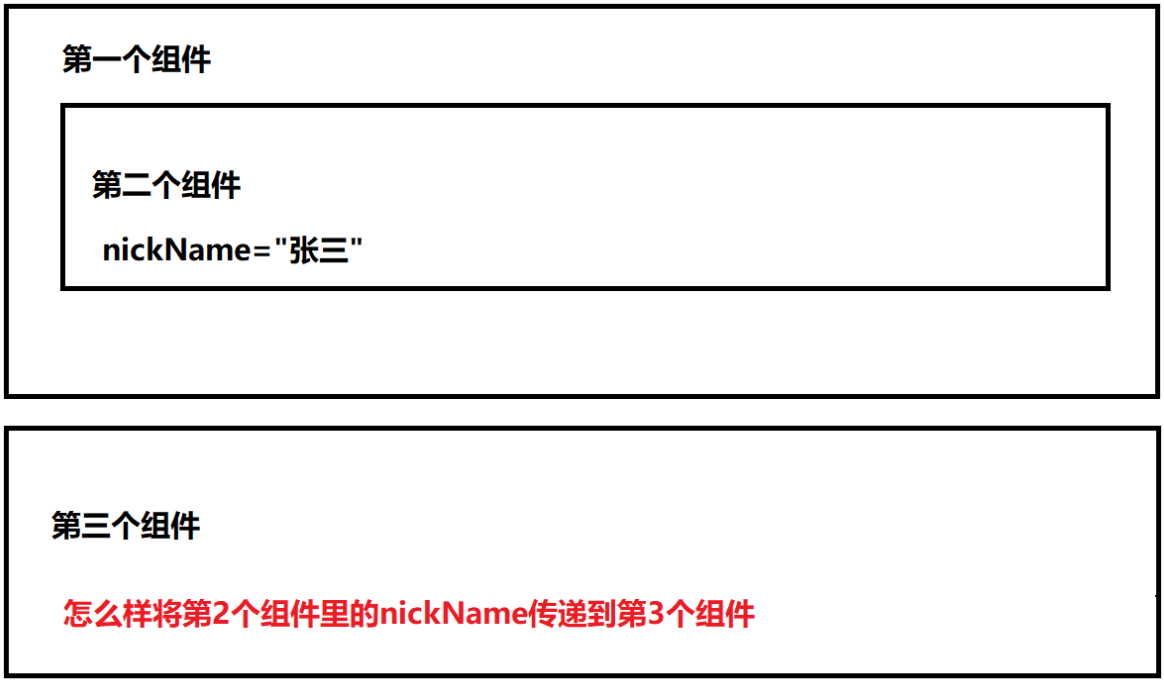
vuex全局状态管理

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前的传值 ，我们有以下的几种方式

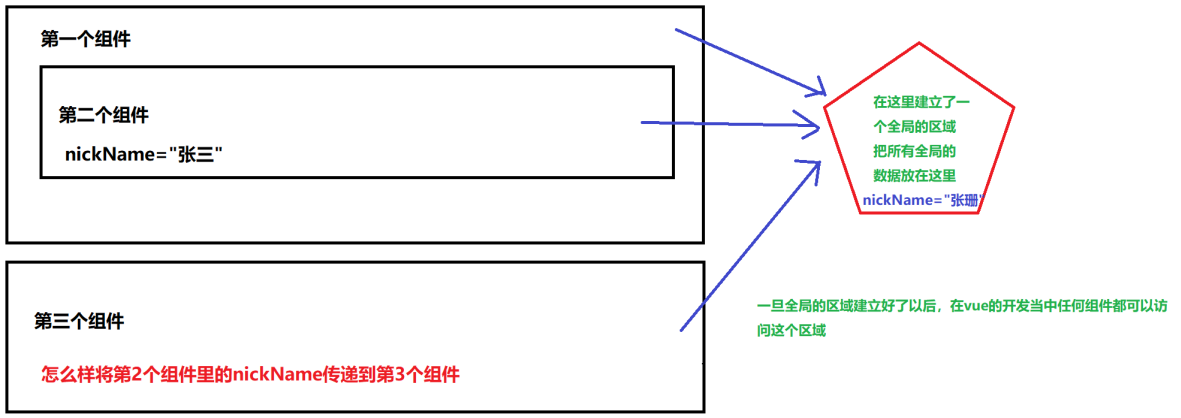
1. 父级组件向子级组件传值使用 `props`
2. 子级组件向父级组件传值使用，插槽， `emit` 自定义事件
3. 跨级组件传值使用 `provide/inject` 【vue3版本支持,这一种方案实现的全局数据因为要遵守数据流的单向性，所以子级组件不能改变全局注入进来的值】



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

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

如果使用的是 `script` 标签引入的，可以直接导入

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

创建store

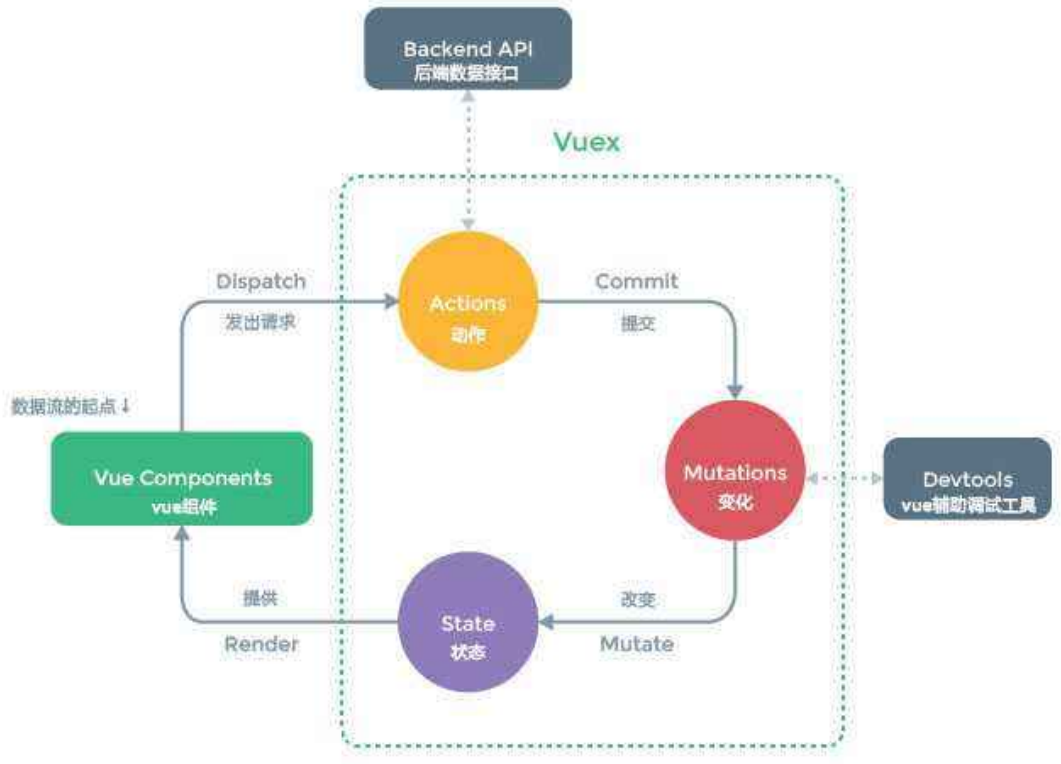
```
//第一步：先创建了vuex的store
const store = Vuex.createStore({

});
```

加载store

```
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

```
state: {
    nickName: "张珊"
}
```

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

```
//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，
//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
mutations:{
    setNickName(state,nickName){
        state.nickName = nickName;
    }
}
```

3.actions方法

```
//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
//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
//我们可以从这个context里面解构出一上方法叫`comment`
actions:{
    setNickName({commit},nickName){
        commit("setNickName",nickName)
    }
}
```

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

```
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`

```
getters:{
    money(){
        return ~~(Math.random()*1000);
    }
}
```

5.vuex的总结

```
  //第一步：先创建了vuex的store
 const store = Vuex.createStore({
     // 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
     state: {
         nickName: "张珊",
         stuList:[
             {stuName:"张三",money:1000},
             {stuName:"李四",money:2000}
         ]
     },
     //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,
     //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
     mutations:{
         setNickName(state,nickName){
             console.log("我是mutations,我触发了");
             state.nickName = nickName;
         }
     },
     //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
     //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
     //我们可以从这个context里面解构出一上方法叫`comment`
     actions:{
         setNickName({commit},nickName){
             console.log("我是action,我触发了");
             commit("setNickName",nickName)
         }
     },
     //这个东西相当于vue里面的计算属性 computed
     //在所有getters的方法里面，系统会注入第一个参数state
```

```
    getters:{
        money(state){
            return ~~(Math.random()*1000);
        },
        totalMoney(state){
            let sum = 0;
            state.stuList.forEach(item=>{
                sum+=item.money;
            });
            return sum;
        }
    }
});
```

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

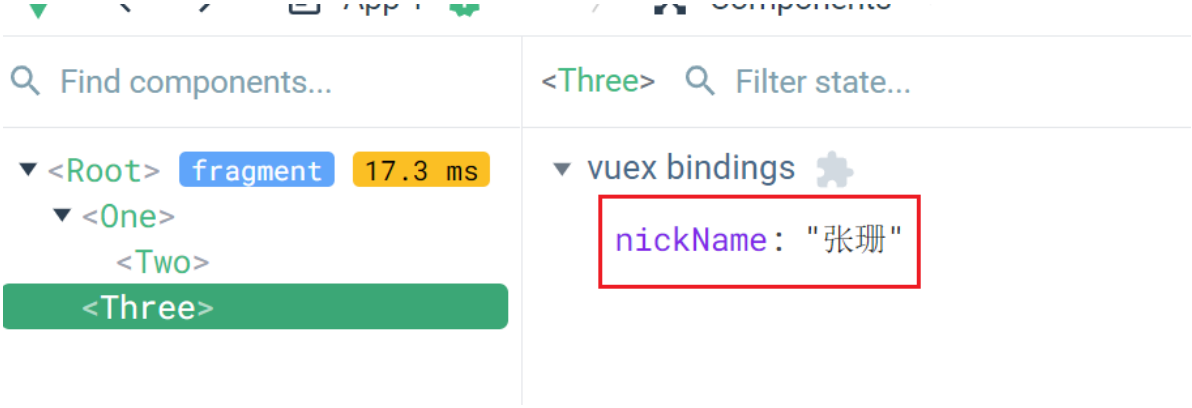
用于将state的值快速的绑定到 `computed`

对象语法

```
app.component("one", {
    template: "#temp1",
    computed:{
        ...mapState({
            nickName:state=>state.nickName
        })
    }
});
```

数组语法

```
app.component("three", {
    template: "#temp3",
    computed:{
        ...mapState(["nickName"])
    }
});
```



这个时候就可以直接使用 `nickName` 了

```
<h3>昵称：{{nickName}}</h3>
```

2.mapGettters

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

```
computed:{
    ...mapState({
        nickName:state=>state.nickName
    }),
   ...mapGetters(["money","totalMoney"])
}
```

> 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

```
methods:{
    ...mapActions(["setNickName"]),
        changeNickName(){
        // this.$store.dispatch("setNickName","樱木花道");
        this.setNickName("流川枫");
    }
}
```

最终代码

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>vuex的介绍</title>
    <style>
        .box {
```

```
            border: 2px solid black;
            padding: 4px;
            margin: 4px;
        }
    </style>
</head>

<body>
    <div id="app">
        <one>
            <two></two>
        </one>
        <three></three>
    </div>
    <template id="temp1">
        <div class="box">
            <h2>第1个组件</h2>
            <h3>昵称：{{$store.state.nickName}}</h3>
            <h3>金额：{{$store.getters.money}}</h3>
            <h3>金额：{{money}}</h3>
            <h3>总金额：{{totalMoney}}</h3>
            <h3>昵称：{{nickName}}</h3>
            <slot></slot>
        </div>
    </template>
    <template id="temp2">
        <div class="box">
            <h2>第2个组件</h2>
            <button type="button" @click="changeNickName">我要改变
nickName</button>
        </div>
    </template>
    <template id="temp3">
        <div class="box">
            <h2>第3个组件</h2>
            <h3>昵称：{{nickName}}</h3>
        </div>
    </template>
</body>
<script src="./js/vue.global.js"></script>
<script src="./js/vuex.global.js"></script>
<script>
    // 如果想实现vuex的快速操作，一定要借用于它内部的工具
    const {mapState,mapGetters,mapActions} = Vuex;
    const app = Vue.createApp({
        created() {
            console.log(this.$store.state.nickName);
        }
    });

    const store = Vuex.createStore({
        state: {
            nickName: "张珊",
            stuList:[
                {stuName:"张三",money:1000},
                {stuName:"李四",money:2000}
```

```
            ]
        },
        mutations:{
            setNickName(state,nickName){
                state.nickName = nickName;
            }
        },
        actions:{
            setNickName({commit},nickName){
                commit("setNickName",nickName)
            }
        },
        getters:{
            money(state){
                return ~~(Math.random()*1000);
            },
            totalMoney(state){
                let sum = 0;
                state.stuList.forEach(item=>{
                    sum+=item.money;
                });
                return sum;
            }
        }
    });

    app.use(store);

    app.component("one", {
        template: "#temp1",
        computed:{
            ...mapState({
                nickName:state=>state.nickName
            }),
            ...mapGetters(["money","totalMoney"])
        }
    });
    app.component("two", {
        template: "#temp2",
        methods:{
            ...mapActions(["setNickName"]),
            changeNickName(){
                // this.$store.dispatch("setNickName","樱木花道");
                this.setNickName("流川枫");
            }
        }
    });
    app.component("three", {
        template: "#temp3",
        computed:{
            ...mapState(["nickName"])
        }
    });

    app.mount("#app");
</script>
```

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

六、vuex状态持久化

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