插件推荐：

1.Vue 3 snippets 作者：hollowtree

watch监视属性 概念：

1.当被监视的属性变化时，回调函数自动调用，进行相关操作。

2.监视的属性必须存在，才能进行监视

3.监视的两种写法

(1)new Vue 时传入`watch`配置

(2)通过 `vm.$wathch` 监视

new Vue 时传入`watch`配置

在初始化时，让 handler 函数调用一次

`immediate:true`

```
    <div id="a">
        <h2>{{info}}</h2>
        <button @click="change">切换</button>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            data:{
                isHot:true
            },
            computed:{
                info(){
                    return this.isHot ? '热' : '晾';
                }
            },
            methods: {
                change(){
                    this.isHot = !this.isHot;
                }
            },
            watch:{
                // 监视 isHot，不仅可以监视 data中的对象，也可以监视比如 computed、
methods 里面的属性
                isHot:{
                    // 初始化时，让 handler 函数调用一下
                    immediate:true,
```

```
                    // handler是一个函数 什么时候调用？ 当isHot发生改变时，handler有两个参数，newVlaue,oldValue
                    handler(newVlaue,oldValue){
                        console.log('isHot被调用了',newVlaue,oldValue);
                    }
                }
            }
        })
    </script>
```

通过 `vm.$watch` 监视

```
    <div id="a">
        <h2>{{info}}</h2>
        <button @click="change">切换</button>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            data:{
                isHot:true
            },
            computed:{
                info(){
                    return this.isHot ? '热' : '晾';
                }
            },
            methods: {
                change(){
                    this.isHot = !this.isHot;
                }
            },
        })

        vm.$watch('isHot',{
            immediate:true,
            handler(newVlaue,oldValue){
                console.log('isHot被调用了',newVlaue,oldValue);
            }
        })
    </script>
```

深度监视

1. Vue中的watch默认不检测对象内部值的改变(一层)
2. 配置 deep:true 可以检测对象内部值改变(多层)
    1. 备注
        1. Vue 自身可以监测对象内部值的改变，但 Vue 提供的watch默认不可以

2. 使用watch时根据数据的具体结构，决定是否采用深度监视

监视多级结构中，某一个属性的变化。

```
<div id="a">
    <h2>a的值是{{numbers.a}}</h2>
    <button @click="add()">加</button>
</div>

<script>
    const vm = new Vue({
        el:"#a",
        data:{
            numbers:{
                a:1,
                b:1
            }
        },
        methods: {
            add(){
                this.numbers.a++ ;
            }
        },
        watch:{
            // 监视多级结构中，某一个属性的变化
            'numbers.a':{
                handler(newVlaue){
                    console.log('a改变了',newVlaue);
                }
            }
        }
    })
</script>
```

监视多级结构中，所有的属性的变化

`deep:true`

```
<div id="a">
    <h2>a的值是{{numbers.a}}</h2>
    <button @click="adda()">加</button>
    <h2>a的值是{{numbers.b}}</h2>
    <button @click="addb()">加</button>
</div>

<script>
    const vm = new Vue({
        el:"#a",
        data:{
            numbers:{
                a:1,
```

```
                b:1
            }
        },
        methods: {
            adda(){
                this.numbers.a++ ;
            },
            addb(){
                this.numbers.b++ ;
            }
        },
        watch:{
            numbers:{
                // 深度检测的一个配置 deep
                deep:true,
                handler(newVlaue){
                    console.log('number中的属性发生了变化了',newVlaue);
                }
            }
        }
    })
</script>
```

简写形式

内部简写

内部`watch`配置时，当只有`handler`函数时，可以使用简写形式，如果有配置项`immediate:true`、`deep:true`的话，不可以使用简写形式

```
<div id="a">
    <h3>{{info}}</h3>
    <button @click="change">按钮</button>
</div>

<script>
    const vm = new Vue({
        el:"#a",
        data:{
            isHot:true
        },
        computed: {
            info(){
                return this.isHot ? '热' :'晾';
            }
        },
        methods: {
            change(){
                this.isHot = !this.isHot
            }
        },
```

```
            watch:{
                // 简写形式,当只有 handler函数时，可以使用简写形式
                isHot(newVlaue,oldValue){
                    console.log('isHot被修改了',newVlaue,oldValue);
                }
            }
        })
    </script>
```

外部简写

通过`vm.$watch`外部简写，当只有`handler`函数时，可以使用简写形式，如果有配置项`immediate:true` 、 `deep:true`的话，不可以使用简写形式

```
    <div id="a">
        <h3>{{info}}</h3>
        <button @click="change">按钮</button>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            data:{
                isHot:true
            },
            computed: {
                info(){
                    return this.isHot ? '热' :'晾';
                }
            },
            methods: {
                change(){
                    this.isHot = !this.isHot
                }
            }
        })
        vm.$watch('isHot',function(newVlaue,oldValue){
            console.log('isHot被调用了',newVlaue,oldValue);
        })
    </script
```

`computed`和`watch`之间的区别

1. `computed`能完成的功能，`watch`都可以完成
2. `watch`能完成的功能，`computed`不一定能完成，例如：`watch`可以进行异步操作

两个重要的小原则：

1. 所有Vue管理的函数，最好写成普通函数，这样 this 的指向才是 vm 或 组件实例对象
2. 所有不被Vue管理的函数(定时器的回调函数，ajax 的回调函数等、Promise的回调函数)，最好写成 箭头函数。

这是 this 的指向 才是 vm 或组件实例对象

```
<div id="root">
   <input type="text" v-model="firstname">
   <br>
   <br>
   <input type="text" v-model="lastname">
   <h2>{{fullName}}</h2>
</div>

<script>
   new Vue({
       el:'#root',
       data:{
           firstname:"张",
           lastname:'三',
           fullName:'张三'
       },
       watch:{
           firstname(newVlaue){
               setTimeout(() => {
                   this.fullName = newVlaue + '-'+this.lastname;
               }, 1000);

           },
           lastname(newValue){
               this.fullName = this.firstname + '-' + newValue;
           }
       }
   })
</script>
```