05.事件和计算属性

学习要点：

1.简单事件

2.计算属性

本节课我们来开始学习 Vue3.x 的事件的入门和计算属性的用法；

一. 简单事件

1. 上节课，我们最后谈到了一个新指令：v-on:click，点击触发事件；

```
<button v-on:click="counter++">+counter</button>
<button v-on:click="counter+=2">+counter</button>
```

PS：当我们点击了按钮即可实现累加 1，或累加 2 的操作；

2. 如果触发的事件业务逻辑较多，那么行内表达式是不够的，需要触发固定方法；

```
<button v-on:click="addCounter">counter+methods</button>
```

```
const App = {
    // 初始数据
    data() {},

    // 事件处理方法
    methods : {
        // 累加方法
        addCounter() {
            // 这里的 this 就是 vue 代理对象
            this.counter++
        }
    }
}
```

PS：所有的方法均可放在 methods 属性对象里，提供给 v-on 触发；

二. 计算属性

1. 对于插值可以进行简单运算，可一旦过于复杂，则模版维护将变得异常困难；

```
<!--两个插值拼装-->
{{firstName + lastName}}
```

```
firstName : 'Mr.',
lastName : 'Lee'
```

2. 上面的例子出现要使用两个插值，如果假设插值内部还要各种运算，极不方便；
3. 此时，我们可以用第二套方案，就是使用方法进行返回，把运算和显示整合起来；

```
<!--直接调用方法，但需要括号-->
{{fullName()}}

methods : {
    // 连接两个属性字符串
    fullName() {
        return this.firstName + this.lastName
    }
}
```

PS：第二套方案在复杂性高的情况下优于第一种，并且插值只调用一次即可；
PS：缺点也很明显，插值调用必须方法模式(有括号)，和属性方式容易混淆；
PS：其次，每次都必须执行方法，没有缓存会在复杂的情况下影响性能；

4. 那么，系统提供了第三种方法解决了这两个问题，就是计算属性；

```
<!--计算属性调用方法-->
{{fullName2}}

const App = {
    // 初始数据
    data() {},

    // 事件处理方法
    methods : {},

    // 计算属性
    computed : {
        // 连接三个属性字符串
        fullName2() {
            return this.firstName + this.LastName
        }
    }
}
```

PS：计算属性和属性一样的调用方式，而方法调用却需要括号，所以推荐用计算属性；
PS：计算属性具有缓存，当值没有改变时，不会重新执行方法，而去使用缓存；
PS：可以参考手册去了解一下计算属性缓存的相关细节，本人推荐需要在实战中体会更好；