04. 计算属性和 setter

学习要点：

1.计算属性

2.setter

本节课我们来开始学习 Vue 的计算属性的用法以及 setter 功能。

一. 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；

```
<!--三个插值拼装-->
<div id="app">
    {{start}} {{message}} {{end}}
</div>

const dataObj = {
    start : '[',
    message : 'Hello, Vue!',
    end : ']',
};
```

2. 上面的例子出现要使用三个插值，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；

```
<!--插值调用方法-->
<div id="app">
    {{welcome()}}
</div>

methods : {
    welcome() {
        console.log('执行了方法' + Math.random());
        return this.start + this.message + this.end;
    }
},
```

4. 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；
5. 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；
6. 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；
7. 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-->
<div id="app">
    {{welcome}}
</div>
```

```
//计算属性
computed : {
    //创建一个方法
    welcome() {
        console.log('执行了计算属性' + Math.random());
        return this.start + this.message + this.end;
    }
}
```

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；

二. setter

1. 一般方法有 getter(取值)和 setter(赋值)两种，计算属性默认只有 getter；
2. 不过，如果你要强行设置，自然也可以让其拥有 setter 功能；

```
//计算属性
computed : {
    welcome : {
        get() {
                console.log('执行了计算属性' + Math.random());
                return this.start + this.message + this.end;
        },
        set(value) {
            this.start = value.split(',')[0];
            this.end = value.split(',')[1];
            console.log('执行了 setter');
        }
    }
}
```