计算属性的目的是简化模板内的表达式，将一些逻辑复杂的模板内的表达式放入计算属性中。

Eg：

<div >

{{ message,split(‘ ’).reverse.join(‘ ’) }} //反转字符串

</div>

通过计算属性可简化模板语言为

<div>

原来的message：{{ message }}

翻转后的message：{{ reverseMessage }}

</div>

data (){

return{

message:’Hello’

}

},

computed:{

reverseMessage:function(){

return this.message.split(‘ ’).reverse.join(‘ ’)

}

}

reverseMessage的值会依赖message值的改变而改变，当message的值通过props当时获取到时也可以这样使用。

计算属性是基于依赖进行缓存的，只有当依赖的数据发生变化时才会重新求值 （区别于在方法中定义reverseMessage：function（）{}）

我们为什么需要缓存？假设我们有一个性能开销比较大的计算属性 A，它需要遍历一个巨大的数组并做大量的计算。

然后我们可能有其他的计算属性依赖于 A 。如果没有缓存，我们将不可避免的多次执行 A 的 getter！

如果你不希望有缓存，请用方法来替代。

计算属性的setter，一般默认我们只使用计算属性的getter。

computed:{

fullName:{

get:function(){

return this.firstName+’ ’+this.lastName

}

set:function(newValues){

var names = newValue.split(‘ ’)

this.firstName = names[0]

this.lastName = names[names.length-1]

}

}

}

2.计算属性在大多数情况下更合适，但有时也需要一个自定义的侦听器。这就是为什么 Vue 通过 watch 选项提供了一个更通用的方法，

来响应数据的变化。当需要在数据变化时执行异步或开销较大的操作时，这个方式是最有用的。





在这个示例中，使用 watch 选项允许我们执行异步操作 (访问一个 API)，限制我们执行该操作的频率，并在我们得到最终结果前，设置中间状态。这些都是计算属性无法做到的。

除了 watch 选项之外，您还可以使用命令式的 [**vm.$watch API**](https://cn.vuejs.org/v2/api/#vm-watch)。