# vue计算属性和观察者

## 计算属性

计算属性（computed）和普通属性一样是在模板中绑定计算属性的，当data中对应数据发生改变时，计算属性的值也会发生改变。

Vue中模板内使用表达式非常便利，但是设计它们的初衷是用于简单运算的。在模板中放入太多的逻辑会让模板过重且难以维护。例如（本例借鉴于vue官方文档）：

<div id="example">

{{ message.split('').reverse().join('') }}

</div>

在这个地方，模板不再是简单的声明式逻辑。你必须看一段时间才能意识到，这里是想要显示变量 message 的翻转字符串。当你想要在模板中多次引用此处的翻转字符串时，就会更加难以处理。

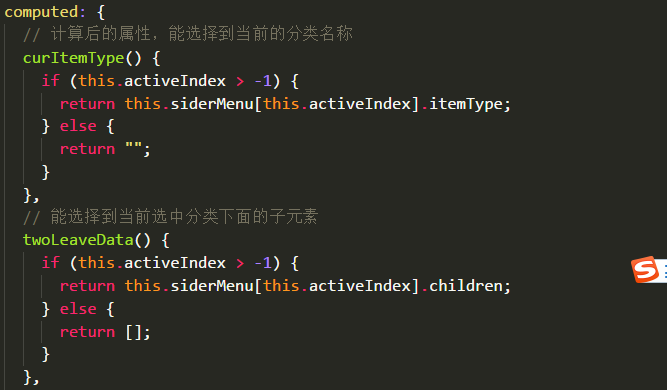
所以，对于任何复杂逻辑，你都应当使用**计算属性**。

## 如何使用计算属性

下面我们结合我们项目，具体介绍计算属性的使用场景，这里我们以首页的商品分类做为分析对象给大家分析下：



首先是当我们经过分类的时候可以返回当前分类的名称，另外还要返回当前分类下面的子类，这个时候就牵扯到比较复杂一些的逻辑处理了，如果我们还是在模板中使用单独的表达式，明显是不合适的，我们就应该去使用计算属性来完成了；下面我们来看下计算属性的使用：



大家看到第一个方法，curItemType可以选择到当前分类的名称，函数内部我们做了判断只有当我们当前索引值大于 - 1的时候才会返回分类名称；

第二个方法twoleaveIndex，能选择到当前选中分类下面的相关热门，这里同理也做了判断，判断方式同上，这里不再重复说明了；

大家如果知道methods方法的话，可以感觉到计算属性（computed）的写法和methods里面函数的写法非常类似，那么这里需要和大家区分一下两者的区别：

## Methods VS computed

methods是方法，只要调用它，函数就会执行。

**相同**：两者达到的效果是同样的。

**不同**：计算属性是基于它们的依赖进行缓存的，只有相关依赖会发生改变时才会重新求值。只要相关依赖未改变，只会返回之前的结果，不再执行函数。

我们这里以curItemType方法为例子给大家做讲解，这里的依赖就是activeIndex，这就意味着只要 activeIndex还没有发生改变，多次访问 curItemType计算属性会立即返回之前的计算结果，而不必再次执行函数。同理如果我们把curItemType方法放在methods里面，每当触发重新渲染时，调用方法将总会再次执行函数。这样的话我们必然会造成不太必要的资源浪费。

这里给大家解释一下缓存的问题，我们为什么需要缓存？假设我们有一个性能开销比较大的的计算属性 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 。

通过以上对于两者异同点的分析，大家可能对于两者的使用场景有了一定的了解，大家在遇到函数的书写时候，可以事先根据以上的一些点自己做判断，然后确定放在的范围；

## 观察者

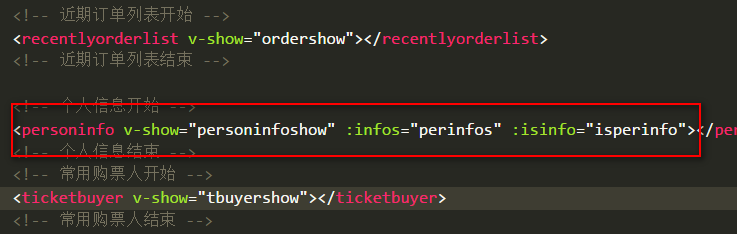
观察者又可以称为侦听属性， Vue 提供了一种更通用的方式来观察和响应 Vue 实例上的数据变动，这就是侦听属性。当你有一些数据需要随着其它数据变动而变动时，我们也可以使用侦听属性。

虽然在大多数情况下计算属性更合适，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。

## 如何使用侦听属性

以我们的项目为例：

这里以我们的个人中心页面为例，个人中心页面中包含了常用购票人页面，这里包含了我们的个人信息页面，首先我需要给大家梳理一下业务流程，我们在使用我们的项目的时候，我们第一次打开个人信息页面的时候，流程应该是，先从后台的数据库中查询个人的信息，看下是否存有个人的信息，如果有的话，应该在个人信息的页面上渲染出来，这是正常的一个业务流程，这里就牵扯到了，从后台获得数据，然后传递到个人信息组件，个人信息组件拿到数据，然后渲染到个人信息页面，这是我们的业务流程；

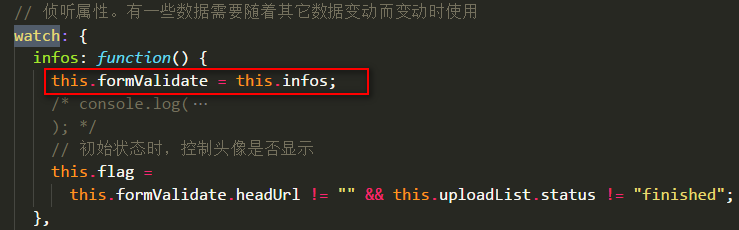


下面我们来说一下业务流程中存在的问题，之前我们的生命周期的文档中我们就有遇到过父子组件的生命周期的问题具体的生命周期排布我们再来看一下：父子组件的执行顺序为，

父组件beforeCreated ->父组件created ->父组件beforeMounted ->子组件beforeCreated ->子组件created ->子组件beforeMounted ->**子组件mounted -> 父组件mounted**。

这里我们看到父组件的mounted函数在子组件mounted之后，那么现在就有一问题了，如果我们在父组件中获得用户的个人信息，然后传递给子组件，但是如果我们在子组件的mounted中去接收父组件传递的数据的话，肯定是接收不到的，因为很明显可以知道父组件的mounted函数在子组件mounted之后，所以是无法接收在子组件的mounted中去接收父组件传递的数据，那么我们怎么解决这个问题，怎么才能接收到这些数据呢？

那么为了解决这个问题，我们就引入了侦听属性（watch属性），下面我看来具体看下代码的实现方式：



我们在父组件的子组件调用上，我们通过infos属性传递从后台获得到得数据，然后在子组件通过props属性进行接收，最后的接收具体操作就是通过侦听属性来操作的，通过监听infos属性，当父组件传递的数据发生变化的时候，这里就会监听到，然后就会回调相应的函数去做处理，看到上面的操作，这里就把传递过来的数据赋值给相应的参数，这样就可以利用参数去渲染显示的页面了；

## Computed VS watch

computed是用于定义基于数据之上的数据。而watch是你想在某个数据变化时做一些事情，如果做的事情是更新其他数据，那其实与把这个要更新的数据项定义成computed是一样的，这个时候用computed更有可读性，虽然技术上讲watch也可以实现。但你也可以在被watch的数据变化时做其他事情，比如调用一个方法，这个是computed做不到也不应该做的。

小结：

1.如果一个数据依赖于其他数据，那么把这个数据设计为computed

2.如果你需要在某个数据变化时做一些事情，就应该去使用watch来观察这个数据变化

## 总结

对于计算属性和观察者属性，在整个Vue框架体系中用的还是比较多的，特别是数据驱动的框架中，大家如果想了解更多的computed和watch属性的相关内容的话，可以去vue官方网站去了解：

地址：<https://cn.vuejs.org/v2/guide/computed.html>