在Vue中什么情况用computed，watch，methods个人理解

> 链接：https://www.jianshu.com/p/709a7bd05da4

用一个例子来说明一下吧：

```
<div id="demo">
  <p>message: "{{ message }}"</p>
  <p>reversed message: "{{ reversedMessage }}"</p>
  <button @click="add">add</button>
  <p>点击次数:{{num}}</p>
</div>

var vm = new Vue({
  el: '#demo',
  data: {
    message : 'abcde',
    num: 0
  },
  watch: {
    // 监听数据的变化做出对应的改变，并不会生成新的属性
    num (newVal) {
        if (newVal > 5) this.alert()
    }
  },
  methods: {
    alert () {
      alert('点击次数达到'+this.num+'次啦')
    },
    add () {
      this.num += 1
    }
  },
  computed: {
    // 根据原有值生成一个新的属性值
    reversedMessage () {
      return this.message.split('').reverse().join('')
    }
  }
})
```

`methods`：存放的方法是一些内部方法、事件的回调、命令是调用的方法。 `watch`：用于监听数据的实时的变化。在数据变化的回调中执行异步操作或者开销很大的时候使用。 `computed`：也是实时监听数据变化，做出相应的变化，跟`watch`不同的是他可以被看成一个`data`里面的属性值来使用。所以当我们需要监听一个值并且需要生成一个新的属性时就可以使用`computed`。

cumputed vs methods

有些时候 `computed` 和 `methods` 可以完成同样的事情，但是 `computed` 是基于它们的依赖进行缓存的，而 `methods` 需要每次去计算。

```
 <p>reversed message: "{{ reversedMessage() }}"</p>

{
  methods: {
    reversedMessage () {
       return this.message.split('').reverse().join('')
    }
  }
}
```

`computed` 只有在他依赖的数据发生变化时才会求值。依赖不发生改变，每次访问 `computed` 应该是之前的计算结果。 也就是说需要缓存的话可以使用 `computed` 来实现，不需要的话可以用其他方式代替。

cumputed vs watch

官方的例子

```
<div id="demo">{{ fullName }}</div>
// 用watch实现
var vm = new Vue({
  el: '#demo',
  data: {
    firstName: 'Foo',
    lastName: 'Bar',
    fullName: 'Foo Bar'
  },
  watch: {
    firstName: function (val) {
      this.fullName = val + ' ' + this.lastName
    },
    lastName: function (val) {
      this.fullName = this.firstName + ' ' + val
    }
  }
})
// 用computed实现
var vm = new Vue({
  el: '#demo',
  data: {
    firstName: 'Foo',
    lastName: 'Bar'
  },
  computed: {
    fullName: function () {
      return this.firstName + ' ' + this.lastName
    }
  }
})
```

很明显 `computed` 实现起来更简洁更方便也更容易理解。 当你有一些数据需要随着其它数据变动而变动时，你很容易滥用 `watch`。然而，通常更好的想法是使用计算属性而不是命令式的 `watch` 回调。

computed 的 setter 和 getter

默认情况下 只有 `getter`，不过在需要时你也可以提供一个 `setter`

```
computed: {
  fullName: {
    // getter
    get: function () {
      return this.firstName + ' ' + this.lastName
    },
    // setter
    set: function (newValue) {
      var names = newValue.split(' ')
      this.firstName = names[0]
      this.lastName = names[names.length - 1]
    }
  }
}
```

现在再运行 `vm.fullName = 'John Doe'` 时， `setter` 会被调用， `vm.firstName` 和 `vm.lastName` 也相应地会被更新。

虽然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 watcher。这是为什么 Vue 通过 watch 选项提供一个更通用的方法，来响应数据的变化。当你想要在数据变化响应时，执行异步操作或开销较大的操作，这是很有用的。