# Vuejs官方文档学习笔记

一：介绍：

Vue 的核心库只关注视图层，另一方面，Vue 完全有能力驱动采用**[单文件组件](http://cn.vuejs.org/v2/guide/single-file-components.html)**和**[Vue生态系统支持的库](http://github.com/vuejs/awesome-vue" \l "libraries--plugins)**开发的复杂单页应用。

Vue.js 的目标是通过尽可能简单的 API 实现****响应的数据绑定****和****组合的视图组件****。

知道 Vue.js 与其它库/框架的区别，查看**[对比其它框架](http://cn.vuejs.org/v2/guide/comparison.html)**。

二：实例化：

1. 实例化 new Vue()
2. 属性代理

三：模板

四：计算属性

1. Computed:{

Todo:function(){

Return “”

}

}

{{todo}}

2. Methods:{

Todo:function(){

Return “”;

}

}

{{todo()}

注意：差别

你可能已经注意到我们可以通过调用表达式中的method来达到同样的效果：

|  |
| --- |
| <p>Reversed message: "{{ reverseMessage() }}"</p> |

|  |
| --- |
| // in component  methods: {  reverseMessage: function () {  return this.message.split('').reverse().join('')  }  } |

不经过计算属性，我们可以在 method 中定义一个相同的函数来替代它。对于最终的结果，两种方式确实是相同的。然而，不同的是****计算属性是基于它的依赖缓存****。计算属性只有在它的相关依赖发生改变时才会重新取值。这就意味着只要 message 没有发生改变，多次访问 reversedMessage 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。

这也同样意味着如下计算属性将不会更新，因为 Date.now() 不是响应式依赖：

|  |
| --- |
| computed: {  now: function () {  return Date.now()  }  } |

相比而言，每当重新渲染的时候，method 调用****总会****执行函数。

我们为什么需要缓存？假设我们有一个重要的计算属性 ****A**** ，这个计算属性需要一个巨大的数组遍历和做大量的计算。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 method 替代。

当 v-bind:style 使用需要特定前缀的 CSS 属性时，如 transform ，Vue.js 会自动侦测并添加相应的前缀。