**注意点：**

**1、不要在选项属性或回调上使用箭头函数**

（Vue实例>3、实例的生命周期）

比如 created: () => console.log(this.a) 或

vm.$watch('a', newValue => this.myMethod())。因为箭头函数是和父级上下文绑定在一起的，this 不会是如你所预期的 Vue 实例，且 this.a 或 this.myMethod 也会是未定义的。

**2、一次性插值**

（模板语法>1、“Mustache”语法）

使用 [v-once 指令](https://cn.vuejs.org/v2/api/#v-once)，执行一次性地插值，当数据改变时，插值处的内容不会更新。但元素/组件及其所有的子节点将被视为静态内容并跳过。

**3、绝不要对用户提供的内容使用HTML插值**

（模板语法>1、“Mustache”语法）

你的站点上动态渲染的任意 HTML 可能会非常危险，因为它很容易导致 [XSS 攻击](https://en.wikipedia.org/wiki/Cross-site_scripting)。请只对可信内容使用 HTML 插值，绝不要对用户提供的内容使用插值。

**4、模板表达式只能访问全局变量的一个白名单**

（模板语法>2、完全的JavaScript表达式支持）

模板表达式都被放在沙盒中，只能访问全局变量的一个白名单，如 Math 和 Date 。你不应该在模板表达式中试图访问用户定义的全局变量。

[**Vue实例**](https://cn.vuejs.org/v2/guide/instance.html)

**1、Vue实例是响应式系统**

Vue实例中的数据和原始数据是双向绑定的，相互影响。

值得注意的是只有当实例被创建时 data 中存在的属性是响应式的。

**2、反向捕获传入Vue实例的对象**

带有前缀 $，以便与用户定义的属性区分开来

[Vue实例的属性和方法](https://cn.vuejs.org/v2/api/#实例属性)

**3、实例的生命周期**

created、mounted、updated、destroyed **（注意点1）**

created: 实例被创建后调用

mounted: el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子（即接管dom后）

updated: 由于数据更改导致的虚拟 DOM 重新渲染和打补丁，在这之后会调用该钩子

[**模板语法**](https://cn.vuejs.org/v2/guide/syntax.html)

**1、常见形式：“Mustache”语法（双大括号）**

双大括号会将数据解释为普通文本，而非HTML代码。（注意点2）

输出真正的 HTML ，你需要使用 v-html 指令。（注意点3）

“Mustache” 语法不能作用在 HTML 特性上，遇到这种情况应该使用 v-bind 指令；

v-bind用于布尔类特许时，如果求值结果是[falsy](https://developer.mozilla.org/zh-CN/docs/Glossary/Falsy)的值，该特性将会被删除。

**2、完全的JavaScript表达式支持**

有个限制就是，每个绑定都只能包含单个表达式，所以下面的例子都不会生效。（注意点4）

<!-- 这是语句，不是表达式 -->

{{ var a = 1 }}

<!-- 流控制也不会生效，请使用三元表达式 -->

~~{{ if (ok) { return message } }}~~

{ok ? 'yes' : 'no'}

**3、参数、修饰符（modifiers）、缩写**

一些指令能够接收一个“参数”，在指令名称之后以冒号表示，如v-bind指令

修饰符 (Modifiers) 是以半角句号 . 指明的特殊后缀，用于指出一个指令应该以特殊方式绑定，如：.prevent修饰符调用event.preventDefault()

v-bind和v-on的缩写：

<!-- 完整语法 -->

<a v-bind:href="url"></a>

<!-- 缩写 -->

<a :href="url"></a>

<!-- 完整语法 -->

<a v-on:click="doSomething"></a>

<!-- 缩写 -->

<a @click="doSomething"></a>

[**计算属性和监听**](https://cn.vuejs.org/v2/guide/computed.html)

**1、计算属性**

插值表达式专注显示，逻辑放在计算属性中



**1.1、计算属性 VS 方法**

**计算属性缓存机制：计算属性是基于它们的依赖进行缓存的**

意味着，只有message变化时，才会执行reversedMessage函数

当需求不希望有缓存时，请在 methods 选项中使用方法

**1.2、计算属性 VS 侦听器**

使用计算属性会更简洁，大多时候能满足需求

当属性是（异步）多状态时，计算属性无法满足，请 watch 选项中使用侦听属性

**1.3、计算属性的getter和setter**

计算属性默认只有 getter ，不过在需要时你也可以提供一个 setter，将



**2、侦听器**

https://cn.vuejs.org/v2/guide/computed.html#%E4%BE%A6%E5%90%AC%E5%99%A8

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。

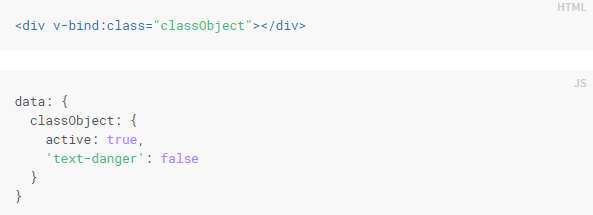
[**Class与Style绑定**](https://cn.vuejs.org/v2/guide/class-and-style.html)

**1、对象语法**

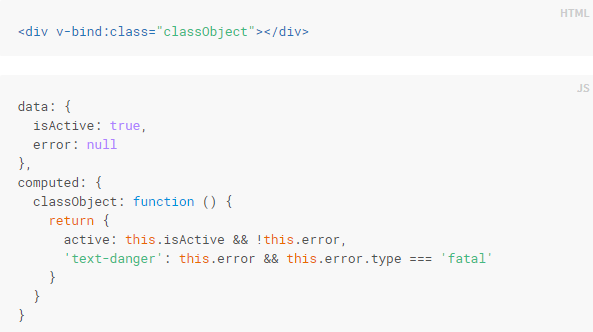
可以是字面量对象



对象名



可以绑定为计算属性



**2、数组语法**

应用于一个class列表，class最终渲染为属性值



在数组语法中也可以嵌套使用对象语法

clipboard.png

**3、用在组件上**

已存在的类不会被覆盖



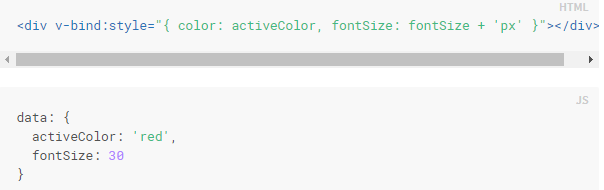
应用对象语法或者数组语法



**4、内联样式的绑定语法和class的一样**

**注意：使用对象字面量时，css属性名可以使用驼峰式（camelCase）,**

**也可以使用短横线分割（kebeb-case, 不过这时候css属性名需要用单引号括起来）**



可以写成：<div :style="{color: activeColor, **'font-size': fontSize + 'px'**}"></div>

**5、自动添加前缀，多重值（2.3.0+）**

当 v-bind:style 使用需要添加浏览器引擎前缀的 CSS 属性时，如 transform，Vue.js 会自动侦测并添加相应的前缀。



[**条件渲染**](https://cn.vuejs.org/v2/guide/conditional.html)

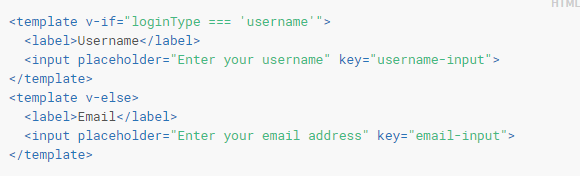
**1、在<template>元素上使用v-if条件渲染分组（多个元素标签）**



**2、v-if中使用key管理可复用元素**

Vue 会尽可能高效地渲染元素，通常会复用已有元素而不是从头开始渲染。

当复用元素不适应需求时，为元素添加一个具有唯一值的key，就可以保证元素的独立性



上述代码，label元素仍会被高效的复用，input相互独立

**3、v-if VS v-show**

v-if 是“真正”的条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建。

v-if 也是惰性的：初始fasly不渲染——直到条件变为Truthy ，才会开始渲染条件块。

**v-show**——不管初始条件是什么，元素总是会被渲染，并且只是**简单地基于 CSS 进行切换**。

相比而已，v-if有更高的切换开销，而v-show有更高的初始渲染开销

因此，**如果需要非常频繁地切换，则使用 v-show 较好**；**如果在运行时条件很少改变，则使用 v-if 较好**。

**3、不推荐v-if和v-for一起使用**

永远不要把 v-if 和 v-for 同时用在同一个元素上。

[**列表渲染**](https://cn.vuejs.org/v2/guide/list.html#v-for-with-v-if)