# **(最好用 Vue.js v1.0.25版本，不要用2.0以上)**

# **Vue.js的常用指令**

用到的v-model是Vue.js常用的一个指令，那么指令是什么呢？

**Vue.js的指令是以v-开头的，它们作用于HTML元素，指令提供了一些特殊的特性，将指令绑定在元素上时，指令会为绑定的目标元素添加一些特殊的行为，我们可以将指令看作特殊的HTML特性（attribute）。**

Vue.js提供了一些常用的内置指令，接下来我们将介绍以下几个内置指令：

* v-if指令
* v-show指令
* v-else指令
* v-for指令
* v-bind指令
* v-on指令

Vue.js具有良好的扩展性，我们也可以开发一些自定义的指令，后面的文章会介绍自定义指令。

## **v-if指令**

v-if是条件渲染指令，它根据表达式的真假来删除和插入元素，它的基本语法如下：

v-if="expression"

expression是一个返回bool值的表达式，表达式可以是一个bool属性，也可以是一个返回bool的运算式。例如：(参考v-if.html页面)

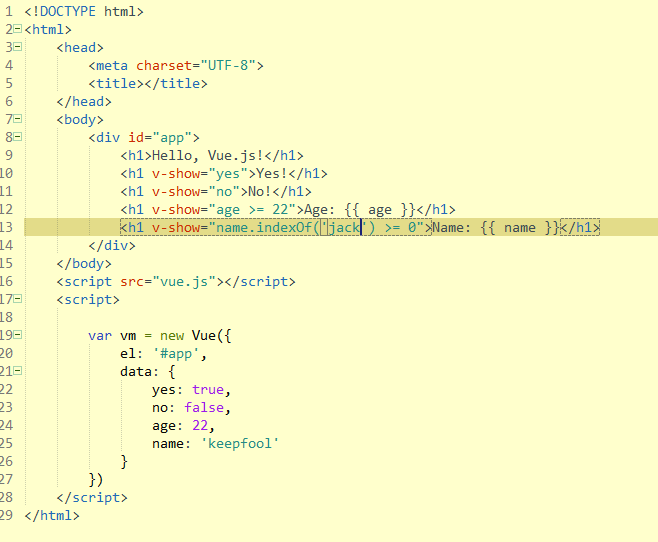


这个例子演示了我们不仅可以绑定 DOM 文本到数据，也可以绑定 DOM 结构到数据。而且，Vue.js 也提供一个强大的过渡效果系统，可以在 Vue 插入/删除元素时自动应用过渡效果。

## **v-show指令**

v-show也是条件渲染指令，和v-if指令不同的是，使用v-show指令的元素始终会被渲染到HTML，它只是简单地为元素设置CSS的style属性。

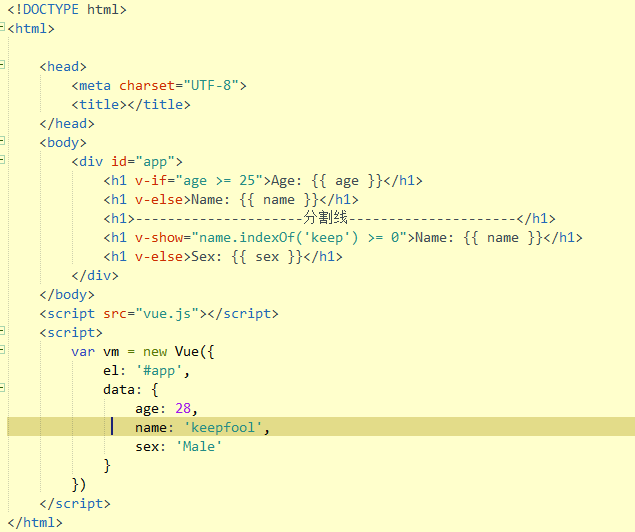
例如：（参考v-show.html）



## **v-else指令**

可以用v-else指令为v-if或v-show添加一个“else块”。v-else元素必须立即跟在v-if或v-show元素的后面——否则它不能被识别。

例如：（参考v-else.html）



## **v-for指令**

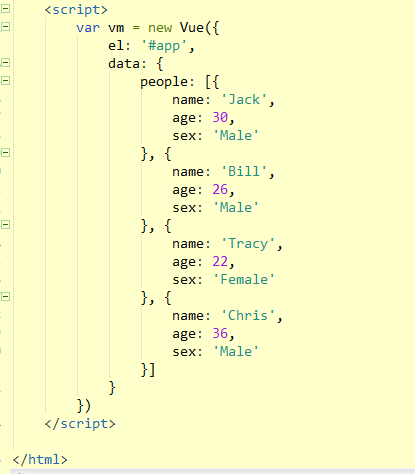
v-for指令基于一个数组渲染一个列表，它和JavaScript的遍历语法相似：

v-for="item in items"

items是一个数组，item是当前被遍历的数组元素。

例如：（参考v-for.html）





我们在选项对象的data属性中定义了一个people数组，然后在#app元素内使用v-for遍历people数组，输出每个person对象的姓名、年龄和性别。

## **v-bind指令**

v-bind指令可以在其名称后面带一个参数，中间放一个冒号隔开，这个参数通常是HTML元素的特性（attribute），例如：v-bind:class

v-bind:argument="expression"

下面这段代码构建了一个简单的分页条，v-bind指令作用于元素的class特性上。  
这个指令包含一个表达式，表达式的含义是：高亮当前页。

例如：（参考v-bind.html）



注意v-for="n in pageCount"这行代码，pageCount是一个整数，遍历时n从0开始，然后遍历到pageCount –1结束。

除了绑定插入的文本内容，我们还可以采用这样的方式绑定 DOM 元素属性：

例如：（参考v-bind1.html）



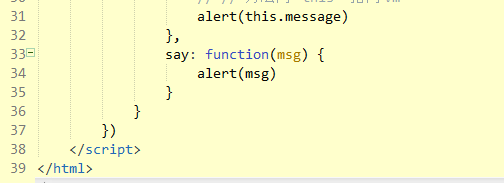
## **v-on指令**

v-on指令用于给监听DOM事件，它的用语法和v-bind是类似的，例如监听<a>元素的点击事件：

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

有两种形式调用方法：**绑定一个方法（让事件指向方法的引用），或者使用内联语句。**Greet按钮将它的单击事件直接绑定到greet()方法，而Hi按钮则是调用say()方法。





## **v-bind和v-on的缩写**

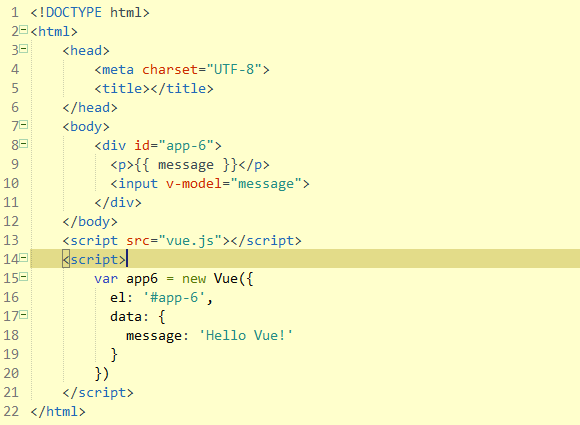
Vue.js为最常用的两个指令v-bind和v-on提供了缩写方式。**v-bind指令可以缩写为一个冒号，v-on指令可以缩写为@符号。**

<!--完整语法--><a href="javascripit:void(0)" v-bind:class="activeNumber === n + 1 ? 'active' : ''">{{ n + 1 }}</a><!--缩写语法--><a href="javascripit:void(0)" :class="activeNumber=== n + 1 ? 'active' : ''">{{ n + 1 }}</a>

<!--完整语法--><button v-on:click="greet">Greet</button><!--缩写语法--><button @click="greet">Greet</button>

**v-model**

Vue 也提供了 v-model 指令，它使得在表单输入和应用状态中做双向数据绑定变得非常轻巧。



# **综合示例**

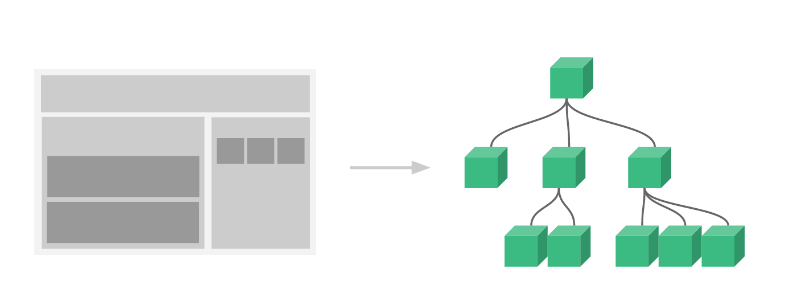
现在我们已经介绍了一些Vue.js的基础知识了，结合以上知识我们可以来做个小Demo。

例如：（参考demo.html）这个页面还有一点点小bug，不过问题也不大

看了上面的文档和案例之后，我们对vue.js应该也有了一定的了解，现在让我们来了解一下用**组件构建（应用）**

**用组件构建（应用）**

组件系统是 Vue.js 另一个重要概念，因为它提供了一种抽象，让我们可以用独立可复用的小组件来构建大型应用。如果我们考虑到这点，几乎任意类型的应用的界面都可以抽象为一个组件树：



在 Vue 里，一个组件实质上是一个拥有预定义选项的一个 Vue 实例：

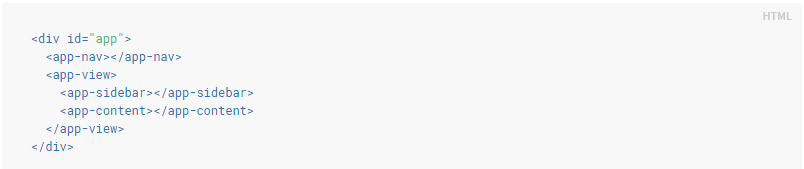




详情可参考http://cn.vuejs.org/v2/guide/#用组件构建（应用）

这只是一个假设的例子，但是我们已经将应用分割成了两个更小的单元，子元素通过 props 接口实现了与父亲元素很好的解耦。我们现在可以在不影响到父应用的基础上，进一步为我们的 todo 组件改进更多复杂的模板和逻辑。

在一个大型应用中，为了使得开发过程可控，有必要将应用整体分割成一个个的组件。在后面的教程中我们将详述组件，不过这里有一个（假想）的例子，看看使用了组件的应用模板是什么样的：



基本所有的都是自定义标签，自己定义功能和内容（好牛逼的样子）

**自定义元素**

你可能已经注意到 Vue.js 组件非常类似于自定义元素——它是 Web 组件规范的一部分。实际上 Vue.js 的组件语法参考了该规范。例如 Vue 组件实现了 Slot API 与 is 特性。但是，有几个关键的不同：

1. Web 组件规范仍然远未完成，并且没有浏览器实现。相比之下，Vue.js 组件不需要任何补丁，并且在所有支持的浏览器（IE9 及更高版本）之下表现一致。必要时，Vue.js 组件也可以放在原生自定义元素之内。
2. Vue.js 组件提供了原生自定义元素所不具备的一些重要功能，比如组件间的数据流，自定义事件系统，以及动态的、带特效的组件替换。

**Vue 实例**

**构造器**