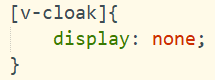
# 基本指令

## v-cloak

v-cloak指令不需要表达式，它会在Vue实例结束编译时从绑定的HTML元素上移除，经常和CSS的 display:none ; 配合使用：



这时虽然已经加了指令v-cloak，但其实并没有起到任何作用，当网速较慢、Vue.js还没加载完时，在页面上会显示 {{message}} 的字样，直到Vue创建实例、编译模板时，DOM才会被替换，所以这个过程屏幕是有闪动的，只要加一句CSS就可以解决这个问题了：



在一般情况下，v-cloak是一个解决初始化慢导致页面闪动的最佳实践，对于简单的项目很实用，但是在具有工程化的项目里，比如后面进阶篇将介绍webpack和vue-router时，项目的HTML结构只有一个空的div元素，剩余的内容都是由路由去挂载不同组件完成的，所以不再需要v-cloak。

## v-once

v-once是一个不需要表达式的指令，被v-once定义的元素或组件只会渲染一次，包括元素或组件的所有子节点，首次渲染后，不再随数据的变化而重新渲染，将被视为静态内容。如下：



v-once在业务中很少使用，当你需要进一步优化性能时，可能会用到。

# 条件渲染指令

## v-if

1. 与JavaScript的条件语句if、else、else if类似，Vue的条件指令可以根据表达式的值在DOM中渲染或销毁元素/组件。例如：



v-else-if要紧跟v-if，v-else要紧跟v-if或v-else-if，v-if也可以单独使用；当表达式的值为真时，当前元素/组件及所有子节点将会被渲染到视图上，为假时，则不渲染。

1. 如果一次判断要显示多个元素，可以在Vue.js内置的<templdate>元素上使用条件指令，最终的渲染结果不会包含<template>元素本身，而是<template>元素中的子元素。例如：



1. Vue在渲染元素时，出于效率考虑，会尽可能地复用已有的元素而非重新渲染。例如：



打开浏览器，键入内容后，点击切换按钮，虽然DOM变了：

切换前：



切换后：



但是之前在输入框输入的内容并没有改变，这说明<input>元素被复用了；如果你不希望这样做，可以使用Vue.js提供的key属性，它可以让你自己决定是否要复用元素，key的值必须是唯一的，例如：



给两个<input>元素都增加key后，就不会复用的情况了，切换类型时键入的内容也会被删除，不过<lable>元素仍然是被复用的，因为没有添加key属性。

## v-show

v-show指令的用法与v-if基本一致，只不过v-show是改变元素的CSS属性display。当v-show表达式的值为false时，元素会隐藏，查看DOM结构会看到元素上加载了内联样式 display:none; 。例如：



渲染后的结果为：



需要注意：v-show不能用在<template>元素上。

## v-if与v-show的比较

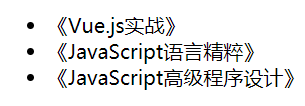
v-if和v-show指令具有类似的功能，不过v-if指令才是真正的条件渲染，它会根据表达式适当的销毁或重建元素及绑定的事件或子组件，若表达式为false，则一开始元素/组件并不会渲染，只有当条件第一次变为真时才会渲染。而v-show只是简单的CSS属性切换，无论条件真与否，都会被渲染，然后再根据v-show表达式判断是否显示该元素/组件。相比之下，v-if更适合条件不经常改变的场景，因为它切换开销较大，而v-show适用于频繁切换条件。

# 列表渲染指令v-for

## 基本用法

1. 当需要将一个数组遍历或枚举一个对象循环显示时，就会用到列表渲染指令v-for，它的表达式需结合in来使用。如下：



 运行结果如下：

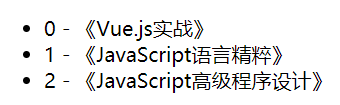
在表达式中，books是数据，book是当前数组元素的别名，循环出的每个<li>内的元素都可以访问到对应的当前数据book。另外，列表渲染也支持用of来代替in作为分隔符：



2）v-for指令的表达式支持一个可选参数作为当前项的索引。如下：



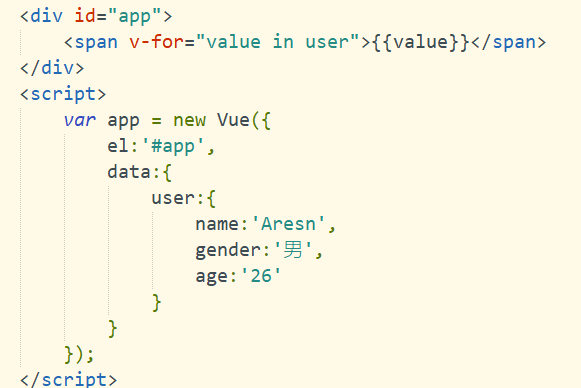
运行结果为：



3）与v-if一样，v-for指令也可以用于<template>元素上，渲染多个元素。如下：



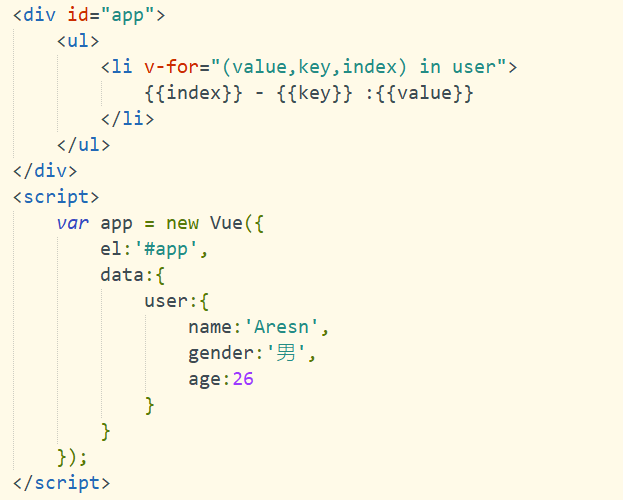
4）v-for指令除了可以遍历数组外，还可以遍历对象的属性。如下：



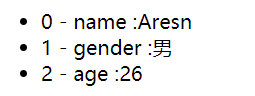
渲染后的结果为：



5）使用v-for遍历对象的属性时，有两个可选参数：键名和索引。如下：



运行结果为：



6）v-for指令除了可以遍历数组或对象，还可以迭代整数。如下：



运行结果如下：



## 数组更新

Vue的核心是数据与视图的双向绑定，当我们修改数组时，Vue会检测到数据变化， 所以用v-for指令渲染的视图也会立即更新。

1. Vue包含了一组观察数组变异的方法，使用这些方法改变数组也会触发视图更新：

·push( ) ·pop( ) ·shift( ) ·unshift( ) ·splice( )

·sort( ) ·reverse( )

例如：



在后面使用了push( )方法向books数组中添加了一个对象，它改变了数组，所以也会触发视图更新。

1. 使用以上的方法会改变被这些方法调用的原始数组，有些方法不会改变原数组：

·filter( ) ·concat( ) ·slice( )

这些方法返回的是一个新数组，在使用这些非变异方法时，可以用新数组来替换原数组，如找出含有JavaScript关键词的书，如下：



渲染结果中，只会显示书名中含有JavaScript的选项。

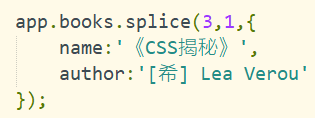
1. 需要注意的是，如果是通过以下方式变动数组，Vue是不能检测到的，所以不会触发视图更新：

·通过索引值直接设置项，比如app.books[3] = { … }

·修改数组长度，比如app.books.length = 1

（1）解决第一种情况有两种方法，第一种方法是使用Vue内置的set方法：



 第二种方法是：

（2）解决第二种情况可以使用使用splice方法解决：



## 过滤和排序

当你不想改变原数组，而是想通过一个数组的副本来做过滤来做过滤或排序的显示时，可以使用计算属性来返回过滤或排序后的数组。

1. 通过计算属性过滤出书名中包含JavaScript关键词的数据，它不会修改原数组：



定义的计算属性computed用于过滤出包含JavaScript词的书名的书。

1. 实现排序也是类似的，定义一个计算属性sortBooks根据书名由长到短进行排序：



# 事件与方法

## 基本用法

Vue.js提供了v-on指令处理事件绑定，类似原生JavaScript的onclick等写法，也是在HTML上进行监听的。

1. 监听一个按钮的单击事件，设置一个计数器，每次点击都加1。如下：



@click是v-on:click的语法糖。

2）@click的表达式可以直接使用JavaScript语句，如上一个案例中，也可以是Vue实例

中methods参数中定义的一个方法。如对上一个案例进行改写：



在Vue实例methods参数中定义了方法供@click调用，需要注意的是，@click调用的方法名后面可以不跟括号“( )”，此时，如果该方法有参数，默认会将原生时间event传入，在大部分业务场景中，如果方法不需要传入参数，为了简便可以不写括号。

这种在HTML元素上监听事件的设计看似将DOM与JavaScript紧密耦合，违背了分离的原理，但实际上刚好相反，因为通过HTML就可以知道调用的是哪个方法，将逻辑与DOM解耦，便于维护；最重要的是，当ViewModel销毁时，所有的事件处

理器都会自动删除，无须自己清理。

1. Vue提供了一个特殊变量@event，用于访问原生DOm事件。例如，使用@event阻止链接打开：



## 修饰符

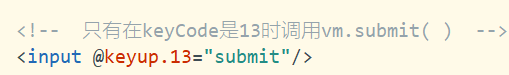
1. 在上例使用的event.preventDefault( )也可以用Vue事件的修饰符来实现，在@绑定的事件后加上小圆点“.”，再跟一个后缀使用修饰符。Vue支持一下修饰符：

·.stop ·.prevent ·capture ·.self ·once

具体用法如下：



1. 在表单元素上监听键盘事件时，还可以使用按键修饰符，比如按下具体某个键时才触发事件，调用方法：



也可以自己配置具体按键：



 全局定义后，使用方式如下：

1. 除了具体的某个keyCode外，Vue还提供了一些快捷名称，以下是全部的别名：

·.enter ·.tab ·.delete（捕获“删除”和“退格”键）

·.esc ·.space ·.up

·.down ·.left ·.right

这些按键修饰符也可以组合使用，或和鼠标一起配置使用：

·.ctrl ·.alt ·.shift

·.meta（Mac下是Command键，Window下是窗口键）

例如：

