Vue.js 使用了基于 HTML 的模板语法，允许开发者声明式地将 DOM 绑定至底层 Vue 实例的数据。所有 Vue.js 的模板都是合法的 HTML ，所以能被遵循规范的浏览器和 HTML 解析器解析。

在底层的实现上，Vue 将模板编译成虚拟 DOM 渲染函数。结合响应系统，Vue 能够智能地计算出最少需要重新渲染多少组件，并把 DOM 操作次数减到最少。

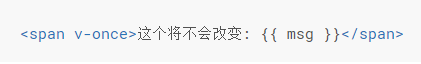
如果你熟悉虚拟 DOM 并且偏爱 JavaScript 的原始力量，你也可以不用模板，[**直接写渲染 (render) 函数**](https://cn.vuejs.org/v2/guide/render-function.html)，使用可选的 JSX 语法。

# 1.数据绑定语法：

## 1.1文本：数据绑定最常见的形式就是使用“Mustache”语法 (双大括号) 的文本插值：



Mustache 标签将会被替代为对应数据对象上 msg 属性的值。无论何时，绑定的数据对象上 msg 属性发生了改变，插值处的内容都会更新。

通过使用 [**v-once 指令**](https://cn.vuejs.org/v2/api/#v-once)，你也能执行一次性地插值，当数据改变时，插值处的内容不会更新。但请留心这会影响到该节点上的其它数据绑定

## 1.2原始html 双大括号会将数据解释为普通文本，而非html代码，想要输出html，需要使用

v-html指令: <span v-html="rawHtml"></span>

rawHtml=’ style=”color:red” ’.

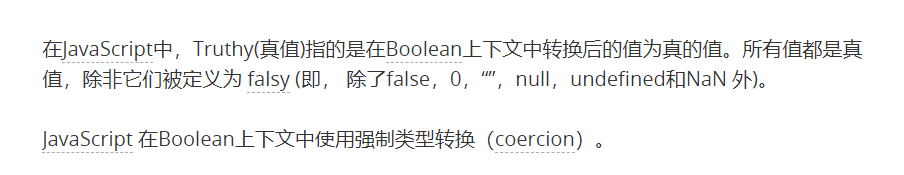
使用了v-html后会忽略解析属性值中的数据绑定。注意：不能用v-html来复核局部模板，因为Vue不是基于字符串的模板引擎。尽量在组件中减少v-html方式来动态渲染界面，这样很容易导致受到XSS攻击

## 1.3 v-bind ----带{{}}的Mustache语法不能使用在html特性（html属性id，calss等）中，遇到这种情况应该使用v-bind（可以使用： 来简化代替）

### 1.3.1带数据绑定的class（对象语法，数组语法，用在组件上）

Eg:（对象语法）

<div v-bind:class=”{ active:isActive }”>//isActive值为 truthy 或false(null,undefine)



<div v-bind:class=”objectClass”> //通过对象的方式传入class

data(){ return{ objectClass:{active:true, ’text-danger’:true } } }

<div v:bind:class=”onjectClass”> //绑定一个返回对象的计算属性

data(){ return{ active:true,error:null } }

computed:{

classObject:function(){

return{

active::this.active&&!this.error,

‘text-danger’:this.error&&this.error.type === ‘fatal’

}

}

}

Eg:（数组语法）

<div v-bind:class=”[objClass1,objClass2]”> //通过给objClass1和 objClass2对象赋值可以动态的改变class的值

一个简单的三元表达式的例子：

<div v-bind:class=”[isActive?objClass1:’ ’,objClass2 ]”> //当isActive值为true时[objClass1,objClass2],为false时[objClass2]

Eg:（用在组件上）

在一个自定义组件上使用class属性后，这些类将会被添加到组件的根元素上边，之后再添加类时已经存在的类不会被覆盖

### 1.3.2绑定内联样式

Eg:（对象语法）



Eg:（多重值）



## 1.4\*\* 使用JavaScript表达式，对于所有的数据绑定，vue都提供了对js表达式的支持



但注意的是，每个绑定的值只能包含单个表达式不能是语句。

# 2.指令

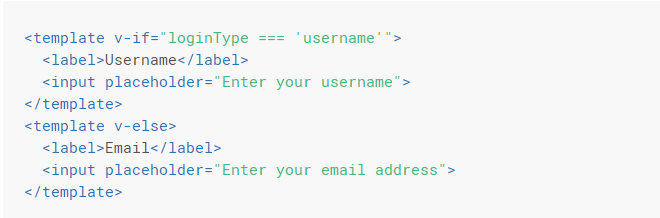
指令是一个带有v-前缀的特殊特性

## 2.1 条件渲染指令v-show与 v-if（v-else || v-else-if）

在template上使用v-if条件渲染分组，可以把template看作是一个不可见的包裹元素，

在上面使用v-if时最终的渲染结果将不包括template。

通过使用Key来管理可复用的元素。Vue会尽可能高效的渲染元素，通常会复用已有元素而不会从头渲染，



在上面的例子中，vue不会重新渲染input，所以你之前输入的信息在切换template的时候还会存在于input框中，这样并不符合我们的预期效果，所以Vue为我们提供了一种两个元素是完全独立的表达方式，只需要添加具有唯一属性的key值就可以了

v-show的用法于v-if相同，唯一不同的是v-show中的元素始终会被渲染并保留在dom中，他只是简单的切换display属性控制显示。V-show不支持template

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*v-if与v-show的区别：\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

v-if是真正的条件渲染，他会确保在 切换过程中，条件快内的事件监听器和子组件 适当的被销毁和重建。v-if也是惰性的，当初始条件为false，他会什么也不做知道第一 个条件变为true

v-show会在任何情况下元素都会被渲染。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 2.2列表渲染（v-for）

v-for需要使用item in items 或 item of items的语法

v-for中我们拥有对父作用域的完全访问权限。

v-for支持两个参数，第一个为item迭代数据别名，第二个为当前项索引index

v-for支持一个对象的v-for，item代表对象的每一个key，对象遍历也支持使用index

（遍历对象的时候是按object.keys()的结果遍历的，不能保证他的结果在不同的js引擎下是相同的）

v-for在元素列表渲染完成后，默认‘就地复用‘策略，当改变列表元素的顺序的时候，v-for不会改变dom的元素顺序，而是简单的复用此处的dom元素。这个模式只适用于不依赖子组件状态或临时dom状态的列表渲染输出。 建议尽可能的在使用v-for时提供key值。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*v-for数组的更新检测\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

简单来说就是当v-for遍历的数组发生变化时，触发视图更新。

一般的当我们使用数组的索引改变数组value的时候(items[0].name=”hhhh”)或使用items.length = newLength方法改变列表的时候不会触发视图的更新，因为vue检测不到这样的数组变动。

### 2.2.1 数组变异方法（会改变原来的数组）

Vue包含一组观察数组变化的变异方法，所以他们会触发视图更新：

push()：push方法可向数组的末尾添加一个或多个元素，并返回新的长度。他会直接修改array而不是创建一个新的数组，返回值为新长度length

pop()：删除并返回数组的最后一个元素，把数组的长度减一，并返回他删除的元素的值，如果数组为空，pop（）不改变数组，并返回undefined

shift()：删除并返回数组的第一个元素并返回原来的第一个值

unshift()：向数组的开头添加一个或多个元素并返回新的数组的长度

splice()：删除元素/向数组添加新的元素，返回被删除的项目，直接对数组进行修改

arrayObject.splice(index,howmany,item1,.....,itemX)

index：必需，整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置

howmany：必须，要删除的项目数量。如果设置为0则不删除项目

item：可选，向数组添加新的项目

sort()：对数组元素进行排序。返回对数组的引用，不生成新的数组

*arrayObject*.sort(*sortby*)

sortby：可选，规定排序顺序，必须是函数

reverse()：颠倒数组中的元素

### 2.2.2.替换数组

也称非变异方法，不改变原来的数组，但是会返回一个新的数组，使用非变异方法时可以用新数组替换旧数组：



concat()：连接两个或更多个数组

slice()：从已有的数组中返回选定的元素。返回一个新的子数组。若果end未规定，选取从start到末尾的所有元素，start为负数表示从尾部选取元素

filter()：创建一个新的数组，新数组中的元素是通过检查指定数组中符合条件的所有元素

### 2.2.3.数组的其他方法

join()：方法用于把数组中的所有元素放入一个字符串。入参表示分隔符，默认为‘,‘

toString():把数组转化为字符串并返回结果

### 2.2.4.set，assign方法（对象属性改变）

Vue会提供一个全局方法set来修改数组或者对象中的值并触发视图更新，

XXX.$set(items,indexOfItems,newValue),

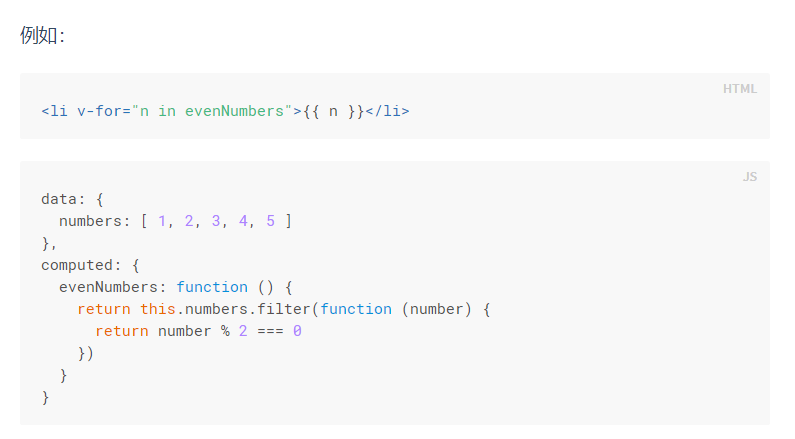
set方法还可以直接添加或删除对象的属性。

当我们需要给对象赋予多个属性时，我们可以使用obj1 = object.assign({},obj2)或者

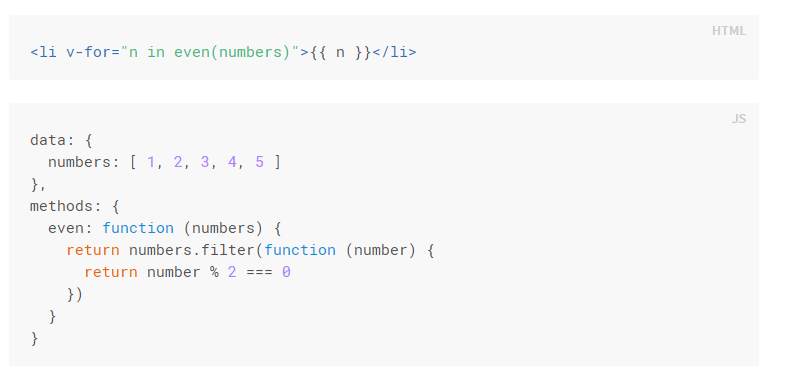
\_.extend()方法

### 2.2.5.显示过滤/排序结果

我们需要一个数组的排序或过滤副本而不改变原来的数组值，我们可以使用conputed计算属性来完成。



在计算属性不适用的时候，我们可以子v-for中嵌套method



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*