

# Vue.js 模板语法

Vue.js 使用了基于 HTML 的模版语法，允许开发者声明式地将 DOM 绑定至底层 Vue 实例的数据。

Vue.js 的核心是一个允许你采用简洁的模板语法来声明式的将数据渲染进 DOM 的系统。

结合响应系统，在应用状态改变时，Vue 能够智能地计算出重新渲染组件的最小代价并应用到 DOM 操作上。

## 插值

### 文本

数据绑定最常见的形式就是使用 `{{...}}`（双大括号）的文本插值：

#### 文本插值

```
<div id="app">
  <p>{{ message }}</p>
</div>
```

[尝试一下 »](#)

## Html

使用 `v-html` 指令用于输出 html 代码：

#### v-html 指令

```
<div id="app">
  <div v-html="message"></div>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      message: '<h1>菜鸟教程</h1>'
    }
  })
</script>
```

[尝试一下 »](#)

## 属性

HTML 属性中的值应使用 `v-bind` 指令。

以下实例判断 `class1` 的值，如果为 `true` 使用 `class1` 类的样式，否则不使用该类：

#### v-bind 指令

```
<div id="app">
  <label for="r1">修改颜色</label><input type="checkbox" v-model="use" id="r1">
```

```
<br><br>
<div v-bind:class="{ 'class1': use }">
v-bind:class 指令
</div>
</div>
<script>
new Vue({
  el: '#app',
  data:{
    use: false
  }
});
</script>
```

[尝试一下 »](#)

## 表达式

Vue.js 都提供了完全的 JavaScript 表达式支持。

### JavaScript 表达式

```
<div id="app">
  {{5+5}}<br>
  {{ ok ? 'YES' : 'NO' }}<br>
  {{ message.split('').reverse().join('') }}
  <div v-bind:id="'list-' + id">菜鸟教程</div>
</div>
<script>
new Vue({
  el: '#app',
  data: {
    ok: true,
    message: 'RUNOOB',
    id : 1
  }
})
</script>
```

[尝试一下 »](#)

## 指令

指令是带有 v- 前缀的特殊属性。

指令用于在表达式的值改变时，将某些行为应用到 DOM 上。如下例子：

### 实例

```
<div id="app">
  <p v-if="seen">现在你看到我了</p>
</div>
<script>
new Vue({
```

```
el: '#app',
data: {
  seen: true
}
})
</script>
```

[尝试一下 »](#)

这里，`v-if` 指令将根据表达式 `seen` 的值(true 或 false )来决定是否插入 `p` 元素。

## 参数

参数在指令后以冒号指明。例如，`v-bind` 指令被用来响应地更新 HTML 属性：

### 实例

```
<div id="app">
<pre><a v-bind:href="url">菜鸟教程</a></pre>
</div>
<script>
new Vue({
  el: '#app',
  data: {
    url: 'http://www.runoob.com'
  }
})
</script>
```

[尝试一下 »](#)

在这里 `href` 是参数，告知 `v-bind` 指令将该元素的 `href` 属性与表达式 `url` 的值绑定。

另一个例子是 `v-on` 指令，它用于监听 DOM 事件：

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

在这里参数是监听的事件名。

## 修饰符

修饰符是以半角句号 `.` 指明的特殊后缀，用于指出一个指令应该以特殊方式绑定。例如，`.prevent` 修饰符告诉 `v-on` 指令对于触发的事件调用 `event.preventDefault()`：

```
<form v-on:submit.prevent="onSubmit"></form>
```

## 用户输入

在 input 输入框中我们可以使用 `v-model` 指令来实现双向数据绑定：

### 双向数据绑定

```
<div id="app">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
<script>
new Vue({
  el: '#app',
  data: {
    message: 'Runoob!'
  }
})
</script>
```

[尝试一下 »](#)

**v-model** 指令用来在 input、select、text、checkbox、radio 等表单控件元素上创建双向数据绑定，根据表单上的值，自动更新绑定的元素的值。

按钮的事件我们可以使用 v-on 监听事件，并对用户的输入进行响应。

以下实例在用户点击按钮后对字符串进行反转操作：

### 字符串反转

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">反转字符串</button>
</div>
<script>
new Vue({
  el: '#app',
  data: {
    message: 'Runoob!'
  },
  methods: {
    reverseMessage: function () {
      this.message = this.message.split('').reverse().join('')
    }
  }
})
</script>
```

[尝试一下 »](#)

## 过滤器

Vue.js 允许你自定义过滤器，被用作一些常见的文本格式化。由"管道符"指示，格式如下：

```
<!-- 在两个大括号中 -->
{{ message | capitalize }}
```

```
<!-- 在 v-bind 指令中 -->
<div v-bind:id="rawId | formatId"></div>
```

过滤器函数接受表达式的值作为第一个参数。

以下实例对输入的字符串第一个字母转为大写：

### 实例

```
<div id="app">
  {{ message | capitalize }}
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      message: 'runoob'
    },
    filters: {
      capitalize: function (value) {
        if (!value) return ''
        value = value.toString()
        return value.charAt(0).toUpperCase() + value.slice(1)
      }
    }
  })
</script>
```

尝试一下 »

过滤器可以串联：

```
{{ message | filterA | filterB }}
```

过滤器是 JavaScript 函数，因此可以接受参数：

```
{{ message | filterA('arg1', arg2) }}
```

这里，message 是第一个参数，字符串 'arg1' 将传给过滤器作为第二个参数，arg2 表达式的值将被求值然后传给过滤器作为第三个参数。

## 缩写

### v-bind 缩写

Vue.js 为两个最为常用的指令提供了特别的缩写：

```
<!-- 完整语法 -->
<a v-bind:href="url"></a>
```


```
<!-- 缩写 -->  
<a :href="url"></a>
```

v-on 缩写


```
<!-- 完整语法 -->  
<a v-on:click="doSomething"></a>  
<!-- 缩写 -->  
<a @click="doSomething"></a>
```

← Vue.js 安装

Vue.js 条件语句 →



3 篇笔记

 写笔记