Vue.js语法

#### v-html

v-html 指令用于输出 html 代码：

<div v-html="message"></div>

<script>

new Vue({ el: '#app', data: { message: '<h1>菜鸟教程</h1>' } })

</script>

#### v-bind

HTML 属性中的值应使用 v-bind 指令。

<div v-bind:class="{'class1': use}"> v-bind:class 指令 </div>

<script> new Vue({ el: '#app', data:{ use: false } }); </script>

缩写：

<!-- 完整语法 -->

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

<!-- 缩写 -->

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

#### 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：

<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>

另一个例子是 v-on 指令，它用于监听 DOM 事件：

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

#### 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：

<div id="app"> <p>{{ message }}</p> <input v-model="message"> </div>

<script> new Vue({ el: '#app', data: { message: 'Runoob!' } }) </script>

**v-model** 指令用来在 input、select、textarea、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：

<div id="app">

<p>{{ message }}</p> <button v-on:click="reverseMessage">反转字符串</button> </div>

<script>

Vue({ el: '#app', data: { message: 'Runoob!' }, methods: { reverseMessage: function () { this.message = this.message.split('').reverse().join('') } } }) </script>

v-on 缩写：

<!-- 完整语法 -->

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

<!-- 缩写 -->

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

#### v-if

条件判断使用 v-if 指令：

<div id="app">

<p v-if="seen">现在你看到我了</p>

<template v-if="ok">

<h1>菜鸟教程</h1>

<p>学的不仅是技术，更是梦想！</p>

<p>哈哈哈，打字辛苦啊！！！</p>

</template>

</div>

<script>

Vue({ el: '#app', data: { seen: true, ok: true } })

</script>

#### v-else

可以用 v-else 指令给 v-if 添加一个 "else" 块：

<div id="app">

<div v-if="Math.random() > 0.5"> Sorry </div> <div v-else> Not sorry </div>

</div>

<script> new Vue({ el: '#app' }) </script>

#### v-else-if

v-else-if 在 2.1.0 新增，顾名思义，用作 v-if 的 else-if 块。可以链式的多次使用

<div id="app">

<div v-if="type === 'A'"> A </div>

<div v-else-if="type === 'B'"> B </div>

<div v-else-if="type === 'C'"> C </div>

<div v-else> Not A/B/C </div>

</div>

<script> new Vue({ el: '#app', data: { type: 'C' } }) </script>

#### v-show

我们也可以使用 v-show 指令来根据条件展示元素：

<h1 v-show="ok">Hello!</h1>

<script>new Vue({el: '#app',data: { ok: true}})</script>

#### v-for

循环使用 v-for 指令。

v-for 指令需要以 **site in sites** 形式的特殊语法， sites 是源数据数组并且 site 是数组元素迭代的别名.

v-for 可以绑定数据到数组来渲染一个列表：

<div id="app">

<ol> <li v-for="site in sites"> {{ site.name }} </li> </ol>

</div>

<script>

new Vue({ el: '#app', data: { sites: [ { name: 'Runoob' }, { name: 'Google' }, { name: 'Taobao' } ] } })

</script>

你也可以提供第二个的参数为键名：

<div id="app">

<ul> <li v-for="(value, key) in object"> {{ key }} : {{ value }} </li> </ul> </div>

第三个参数为索引：

<div id="app">

<ul>

<li v-for="(value, key, index) in object"> {{ index }}. {{ key }} : {{ value }} </li>

</ul>

</div>

#### watch

Vue.js 监听属性 watch，我们可以通过 watch 来响应数据的变化。

<div id = "app">

<p style = "font-size:25px;">计数器: {{ counter }}</p> <button @click = "counter++" style = "font-size:25px;">点我</button>

</div>

<script type = "text/javascript">

var vm = new Vue({ el: '#app', data: { counter: 1 } }); vm.$watch('counter', function(nval, oval) { alert('计数器值的变化 :' + oval + ' 变为 ' + nval + '!'); });

</script>

#### 事件修饰符

Vue.js 为 v-on 提供了事件修饰符来处理 DOM 事件细节，如：event.preventDefault() 或 event.stopPropagation()。

Vue.js 通过由点 **.** 表示的指令后缀来调用修饰符。

* .stop - 阻止冒泡
* .prevent - 阻止默认事件
* .capture - 阻止捕获
* .self - 只监听触发该元素的事件
* .once - 只触发一次
* .left - 左键事件
* .right - 右键事件
* .middle - 中间滚轮事件

<!-- 阻止单击事件冒泡 -->

<a v-on:click.stop="doThis"></a>

<!-- 提交事件不再重载页面 -->

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

<!-- 修饰符可以串联 -->

<a v-on:click.stop.prevent="doThat"></a>

<!-- 只有修饰符 -->

<form v-on:submit.prevent></form>

<!-- 添加事件侦听器时使用事件捕获模式 -->

<div v-on:click.capture="doThis">...</div>

<!-- 只当事件在该元素本身（而不是子元素）触发时触发回调 -->

<div v-on:click.self="doThat">...</div>

<!-- click 事件只能点击一次，2.1.4版本新增 -->

<a v-on:click.once="doThis"></a>

#### 按键修饰符

Vue 允许为 v-on 在监听键盘事件时添加按键修饰符：

<!-- 只有在 keyCode 是 13 时调用 vm.submit() -->

<input v-on:keyup.13="submit">

记住所有的 keyCode 比较困难，所以 Vue 为最常用的按键提供了别名：

<!-- 同上 -->

<input v-on:keyup.enter="submit">

<!-- 缩写语法 -->

<input @keyup.enter="submit">

全部的按键别名：

* .enter
* .tab
* .delete (捕获 "删除" 和 "退格" 键)
* .esc
* .space
* .up
* .down
* .left
* .right
* .ctrl
* .alt
* .shift
* .meta

<p><!-- Alt + C -->

<input @keyup.alt.67="clear">

<!-- Ctrl + Click -->

<div @click.ctrl="doSomething">Do something</div>

#### Input修饰符

.lazy

在默认情况下， v-model 在 input 事件中同步输入框的值与数据，但你可以添加一个修饰符 lazy ，从而转变为在 change 事件中同步：

<!-- 在 "change" 而不是 "input" 事件中更新 -->

<input v-model.lazy="msg" >

.number

如果想自动将用户的输入值转为 Number 类型（如果原值的转换结果为 NaN 则返回原值），可以添加一个修饰符 number 给 v-model 来处理输入值：

<input v-model.number="age" type="number">

.trim

如果要自动过滤用户输入的首尾空格，可以添加 trim 修饰符到 v-model 上过滤输入：

<input v-model.trim="msg">

#### 组件Component

注册一个全局组件语法格式如下：

Vue.component(tagName, options)

tagName 为组件名，options 为配置选项。注册后，我们可以使用以下方式来调用组件：

<tagName></tagName>

prop