## V-text 设置文本的标签值

屏幕上有字

描述已自动生成

结果都是 asd! V-text的名称不一定为message可以设置为你想要的

每个 Vue 应用都需要通过实例化 Vue 来实现。

语法格式如下：

var vm = new Vue({

// 选项

})

<div id="vue\_det">

<h1>site : {{site}}</h1>

<h1>url : {{url}}</h1>

<h1>{{details()}}</h1>

</div>

<script type="text/javascript">

var vm = new Vue({

el: '#vue\_det',

data: {

site: "菜鸟教程",

url: "www.runoob.com",

alexa: "10000"

},

methods: {

details: function() {

return this.site + " - 学的不仅是技术，更是梦想！";

}

}

})

可以看到在 Vue 构造器中有一个el 参数，它是 DOM 元素中的 id。在上面实例中 id 为 vue\_det，在 div 元素中：

<div id = "vue\_det"></div>

这意味着我们接下来的改动全部在以上指定的 div 内，div 外部不受影响。

接下来我们看看如何定义数据对象。

**data** 用于定义属性，实例中有三个属性分别为：site、url、alexa。

**methods** 用于定义的函数，可以通过 return 来返回函数值。

**{{ }}** 用于输出对象属性和函数返回值。

## 文本插值

<div id="app">

<p>{{ message }}</p>

</div>

<div id="feng" v-html="fsg"></div>

  <script> vue必须大写

        new Vue({

           el:'#feng',

           data:{

               fsg:'fengshangguiniubi'

           }

        })

  </script>

### 属性

HTML 属性中的值应使用 v-bind 指令。

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

<style>

    .class1{

        background-color: blue;

    }

</style>

<body>

   <div id="app">

        <p>点击这里</p>

        <input type="checkbox" v-model="use">

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

   </div>

   <script>

    new Vue({

        el:'#app',

        data:{

            use:false,

        }

    });

   </script>

## 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：

<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>

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：

<body>

<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>

</body>

文本

中度可信度描述已自动生成

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：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

形状

描述已自动生成

**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>

new Vue({

el: '#app',

data: {

message: 'Runoob!'

},

methods: {

reverseMessage:function(){

this.message = this.message.split('').reverse().join('')

}

}

})

</script>

<body>

   <div id="fsg">

       <h1>{{message}}</h1>

       <button v-on:click="finDit">找到你啦</button>

   </div>

   <script>

       new Vue({

           el:'#fsg',

           data:{

               message:"456"

           },

           methods: {

               finDit:function(){

                   this.message=this.message+"123"

               }

           },

       })

   </script>

</body>

图形用户界面

中度可信度描述已自动生成

## 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：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

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>

## 条件判断

### v-if

条件判断使用 v-if 指令：

<p v-if="seen">现在你看到我了</p>

seen: true,

### v-else

可以用 v-else 指令给 v-if 添加一个 "else" 块：

<div v-if="Math.random() > 0.5"> Sorry</div>

<div v-else> Not sorry </div>

new Vue({ el: '#app' })

### 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 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>

*v-else 、v-else-if 必须跟在 v-if 或者 v-else-if之后。*

### v-show

我们也可以使用 v-show 指令来根据条件展示元素：

## v-show 指令

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

New vue(){

El:’asd’,

Data:{

Ok:true}

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：

<ol>

<li v-for="site in sites">

{

{ site.name }

} </li>

</ol>

new Vue({

el: '#app',

data: {

sites: [ { name: 'Runoob' },

{ name: 'Google' }, { name: 'Taobao' }

]

}

})

模板中使用 v-for：

**v-for**

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

**v-for 迭代对象**

v-for 可以通过一个对象的属性来迭代数据：

<li v-for="value in object">{{ value }} </li>

new Vue({

el: '#app',

data: {

object: {

name: '菜鸟教程',

url: 'http://www.runoob.com',

slogan: '学的不仅是技术，更是梦想！'

}

}

})

你也可以提供第二个的参数为键名：

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

Key:name url: slogan:

第三个参数为索引：

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

1. name.cainiaojiaocheng
2. v-for 也可以循环整数
3. <li v-for="n in 10"> {{ n }}

计算属性关键词: **computed**。

计算属性在处理一些复杂逻辑时是很有用的。

文本

描述已自动生成

实例 2 中声明了一个计算属性 reversedMessage 。

提供的函数将用作属性 vm.reversedMessage 的 getter 。

vm.reversedMessage 依赖于 vm.message，在 vm.message 发生改变时，vm.reversedMessage 也会更新。

## computed vs methods

图形用户界面, 应用程序

描述已自动生成

我们可以使用 methods 来替代 computed，效果上两个都是一样的，但是 computed 是基于它的依赖缓存，只有相关依赖发生改变时才会重新取值。而使用 methods ，在重新渲染的时候，函数总会重新调用执行。

可以说使用 computed 性能会更好，但是如果你不希望缓存，你可以使用 methods 属性。

# Vue.js 监听属性

以下实例通过使用 watch 实现计数器：

图形用户界面, 文本, 应用程序

描述已自动生成

以下实例进行**千米**与**米**之间的换算：

文本

描述已自动生成