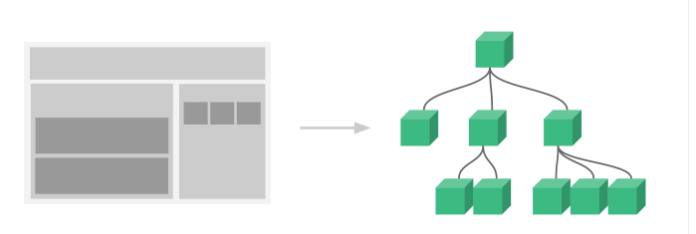
# **[VueJs(8)---组件（注册组件）](https://www.cnblogs.com/qdhxhz/p/8993228.html)**

# **组件（注册组件）**

## **一、介绍**

       组件系统是Vue.js其中一个重要的概念，它提供了一种抽象，让我们可以使用独立可复用的小组件来构建大型应用，任意类型的应用界面都可以抽象为一个组件树



   那么什么是组件呢？

**组件可以扩展HTML元素，封装可重用的HTML代码，我们可以将组件看作自定义的HTML元素。**

## **二、如何注册组件**

   Vue.js的组件的使用有3个步骤：**创建组件构造器、注册组件和使用组件。**



 下面用代码演示这三步

[IMG_258](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<!DOCTYPE html><html>

<body>

<div id="app">

<!-- 注意： #app是Vue实例挂载的元素，应该在挂载元素范围内使用组件-->

<my-component></my-component>

</div>

</body>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>

<!-- 1.创建一个组件构造器 -->

var myComponent = Vue.extend({

template: '<div>This is my first component!</div>'

})

<!-- 2.注册组件，并指定组件的标签，组件的HTML标签为<my-component> -->

Vue.component('my-component', myComponent)

<!-- 3.通过id=app进行挂载 -->

new Vue({

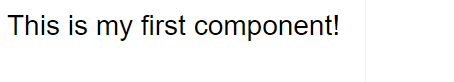
el: '#app'

});

</script></html>

[IMG_259](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

    运行结果如下：



#### 

#### **一、 全局注册和局部注册**

**调用**Vue.component()**注册组件时，组件的注册是全局的，这意味着该组件可以在任意Vue示例下使用。**  
如果不需要全局注册，或者是让组件使用在其它组件内，可以用**选项对象的components属性实现局部注册。**

**我自己的理解只要是component就代表全局组件，components代表局部组件**

    上面的示例可以改为局部注册的方式：

[IMG_261](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<!DOCTYPE html><html>

<body>

<div id="app">

<!-- 3. my-component只能在#app下使用-->

<my-component></my-component>

</div>

</body>

<script src="js/vue.js"></script>

<script>

// 1.创建一个组件构造器

var myComponent = Vue.extend({

template: '<div>This is my first component!</div>'

})

new Vue({

el: '#app',

components: {

// 2. 将myComponent组件注册到Vue实例下

'my-component' : myComponent

}

});

</script></html>

[IMG_262](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

       由于my-component组件是注册在#app元素对应的Vue实例下的，所以它不能在其它Vue实例下使用。

[IMG_263](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<div id="app2">

<!-- 不能使用my-component组件，因为my-component是一个局部组件，它属于#app-->

<my-component></my-component></div>

<script>

new Vue({

el: '#app2'

});</script>

[IMG_264](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

### **二、组件注册语法糖**

    以上组件注册的方式有些繁琐，Vue.js为了简化这个过程，提供了注册语法糖

[IMG_265](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

// 全局注册，my-component1是标签名称

Vue.component('my-component1',{

template: '<div>This is the first component!</div>'

})

var vm1 = new Vue({

el: '#app1'

})

[IMG_266](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

       Vue.component()的第1个参数是标签名称，第2个参数是一个选项对象，使用选项对象的template属性定义组件模板。  
使用这种方式，Vue在背后会自动地调用Vue.extend()。

**components实现局部注册**

[IMG_267](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

var vm2 = new Vue({

el: '#app2',

components: {

// 局部注册，my-component2是标签名称

'my-component2': {

template: '<div>This is the second component!</div>'

},

// 局部注册，my-component3是标签名称

'my-component3': {

template: '<div>This is the third component!</div>'

}

}

})

[IMG_268](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

### **三、父组件和子组件**

 我们可以在组件中定义并使用其他组件，这就构成了父子组件的关系。

[IMG_269](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<!DOCTYPE html><html>

<body>

<div id="app">

<parent-component>

</parent-component>

</div>

</body> <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>

var Child = Vue.extend({

template: '<p>This is a child component!</p>'

})

var Parent = Vue.extend({

// 在Parent组件内使用<child-component>标签 template :'<p>This is a Parent component</p><child-component></child-component>',

components: {

// 局部注册Child组件，该组件只能在Parent组件内使用

'child-component': Child

}

})

// 全局注册Parent组件 Vue.component('parent-component', Parent)

new Vue({

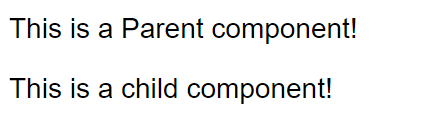
el: '#app'

})

</script></html>

[IMG_270](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

 这段代码的运行结果如下



### **四、使用script或template标签**

       尽管语法糖简化了组件注册，但在template选项中拼接HTML元素比较麻烦，这也导致了HTML和JavaScript的高耦合性。  
庆幸的是，Vue.js提供了两种方式将定义在JavaScript中的HTML模板分离出来。

[IMG_272](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<!DOCTYPE html><html lang="en">

<head>

<meta charset="UTF-8">

<title>vue组件</title>

<script src="js/vue.js"></script></head>

<body>

<div id="app1">

<my-com></my-com>

<my-com1></my-com1>

</div>

<template id="myCom">

<div>这是template标签构建的组件</div>

</template>

<script type="text/x-template" id="myCom1">

<div>这是script标签构建的组件</div>

</script>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>

Vue.component('my-com1', {

template: '#myCom1'

});

var app1 = new Vue({

el: '#app1',

components: {

'my-com': {

template: '#myCom'

}

}

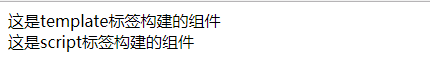
});

</script></body>

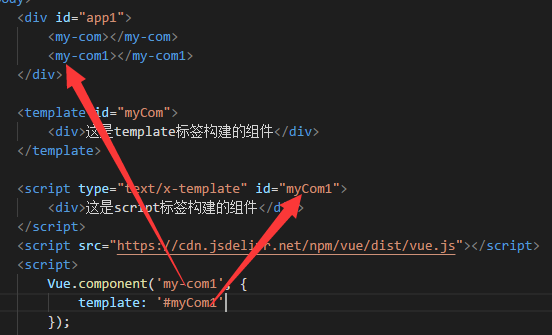
</html>

[IMG_273](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

**运行结果：**

****

**注意：**使用<script>标签时，type指定为text/x-template，意在告诉浏览器这不是一段js脚本，浏览器在解析HTML文档时会忽略<script>标签内定义的内容。



      在理解了组件的创建和注册过程后，我建议使用<script>或<template>标签来定义组件的HTML模板。  
这使得HTML代码和JavaScript代码是分离的，便于阅读和维护。

### **五、模板的注意事项**

     1**. 以子标签的形式在父组件中使用**

<div id="app">

<parent-component>

<child-component></child-component>

</parent-component></div>

 上面是错误的。为什么这种方式无效呢？因为当子组件注册到父组件时，Vue.js会编译好父组件的模板，模板的内容已经决定了父组件将要渲染的HTML。  
<parent-component>…</parent-component>相当于运行时，它的一些子标签只会被当作普通的HTML来执行，<child-component></child-component>不是标准的HTML标签，会被浏览器直接忽视掉

**2.组件的模板只能有一个根元素。下面的情况是不允许的。**

template: `<div>这是一个局部的自定义组件，只能在当前Vue实例中使用</div>

<button>hello</button>`

**3.组件中的data必须是函数**

       注册组件时传入的配置和创建Vue实例差不多，但也有不同，其中一个就是data属性必须是一个函数。

这是因为如果像Vue实例那样，传入一个对象，由于JS中对象类型的变量实际上保存的是对象的引用，所以当存在多个这样的组件时，会共享数据，导致一个组件中数据的改变会引起其他组件数据的改变。

而使用一个返回对象的函数，每次使用组件都会创建一个新的对象，这样就不会出现**共享数据**的问题来了。

**4.关于DOM模板的解析**

       当使用 DOM 作为模版时 (例如，将 el 选项挂载到一个已存在的元素上), 你会受到 HTML 的一些限制，因为 Vue 只有在浏览器解析和标准化 HTML 后才能获取模板内容。尤其像这些元素 <ul>，<ol>，<table>，<select> 限制了能被它包裹的元素，而一些像 <option> 这样的元素只能出现在某些其它元素内部

   在自定义组件中使用这些受限制的元素时会导致一些问题，例如

<table>

<my-row>...</my-row></table>

        自定义组件 <my-row> 被认为是无效的内容，因此在渲染的时候会导致错误。这时应使用特殊的 is 属性：

<table>

<tr is="my-row"></tr></table>

       也就是说，标准HTML中，一些元素中只能放置特定的子元素，另一些元素只能存在于特定的父元素中。比如table中不能放置div，tr的父元素不能div等。所以，当使用自定义标签时，标签名还是那些标签的名字，但是可以在标签的is属性中填写自定义组件的名字。

## **三、动态组件**

    有的时候，在不同组件之间进行动态切换是非常有用的，比如在一个多标签的界面里

    简单点说：就是几个组件放在一个挂载点下，然后根据父组件的某个变量来决定显示哪个，或者都不显示。

**要点：**在挂载点使用**component标签**，然后使用**v-bind:is=”组件名”**，会自动去找匹配的组件名，如果没有，则不显示

动态组件，先看案例效果：

代码演示：css代码就不复制了，上面案例效果里有。

[IMG_276](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<script src="https://unpkg.com/vue"></script>

<div id="dynamic-component-demo" class="demo">

<button v-for="tab in tabs"   
 v-bind:key="tab"   
 v-bind:class="['tab-button', { active: currentTab === tab }]"   
 v-on:click="currentTab = tab">{{ tab }}</button>

<component v-bind:is="currentTabComponent" class="tab"></component></div>

[IMG_277](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

     这里v-bind:key其实可有可无，具体[key](https://cn.vuejs.org/v2/guide/list.html" \l "key" \t "https://www.cnblogs.com/qdhxhz/p/_blank)介绍可以看官网。

     这里v-bind:class和v-on:click都是用来为了改变样式用的。

    关键是component组件标签。

[IMG_278](https://www.cnblogs.com/qdhxhz/p/javascript:void(0);)

<script>

//显示定义了三个组件

Vue.component('tab-科长', {

template: '<div>一共有100个科长</div>'

})

Vue.component('tab-处长', {

template: '<div>一种有50个处长</div>'

})

Vue.component('tab-局长', {

template: '<div>一共有10个局长</div>'

})

new Vue({

el: '#dynamic-component-demo',

data: {

currentTab: '局长',

tabs: ['科长', '处长', '局长']

},

//计算属性，根据currentTab的改变来判断选择哪个组件 computed: {

currentTabComponent: function() {

return 'tab-' + this.currentTab

}

}

})</script>

## **五、扩展实例**

[IMG_256](https://www.cnblogs.com/toward-the-sun/p/javascript:void(0);)

// 定义一个混合对象var myMixin = {

created: function () {

this.hello()

},

methods: {

hello: function () {

console.log('hello from mixin!')

}

}

}// 定义一个使用混合对象的组件var Component = Vue.extend({

mixins: [myMixin]

})var component = new Component() // -> "hello from mixin!"