# Vue组件的知识点：

# 这里有一个 Vue 组件的示例：

// 定义一个名为 button-counter 的新组件

Vue.component('button-counter', {

data: function () {

return {

count: 0

}

},

template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'

})

组件是可复用的 Vue 实例，且带有一个名字：在这个例子中是 <button-counter>。我们可以在一个通过 new Vue 创建的 Vue 根实例中，把这个组件作为自定义元素来使用：

<div id="components-demo">

<button-counter></button-counter>

</div>

new Vue({ el: '#components-demo' })

data必须是一个函数，**一个组件的 data 选项必须是一个函数**

**Vue中有两种组件的注册：全局注册和局部注册**

**Vue.component(‘my-component’,{**

**//此为全局注册**

**})**

全局注册的组件可以用在其被注册之后的任何 (通过 new Vue) 新创建的 Vue 根实例，也包括其组件树中的所有子组件的模板中。

## 2.通过prop向子组件传递数据

Prop 是你可以在组件上注册的一些自定义特性。当一个值传递给一个 prop 特性的时候，它就变成了那个组件实例的一个属性。

Vue.component('blog-post', {

props: ['title'],

template: '<h3>{{ title }}</h3>'

})

当组件变得复杂的时候，也就是所需要的信息更多的时候。

<blog-post

v-for="post in posts"

v-bind:key="post.id"

v-bind:title="post.title"

v-bind:content="post.content"

v-bind:publishedAt="post.publishedAt"

v-bind:comments="post.comments"

></blog-post>

**监听子组件事件：\*\*\***

//注册一个实例

new Vue({

el: '#blog-posts-events-demo',

data: {

posts: [/\* ... \*/],

postFontSize: 1

}

})

//引用模板

<div id="blog-posts-events-demo">

<div :style="{ fontSize: postFontSize + 'em' }">

<blog-post

v-for="post in posts"

v-bind:key="post.id"

v-bind:post="post"

></blog-post>

</div>

</div>

//注册全局组件

Vue.component('blog-post', {

props: ['post'],

template: `

<div class="blog-post">

<h3>{{ post.title }}</h3>

<button>

Enlarge text

</button>

<div v-html="post.content"></div>

</div>

`

})

问题是这个按钮不会做任何事：

<button>

Enlarge text

</button>

当点击这个按钮时，我们需要告诉父级组件放大所有博文的文本。幸好 Vue 实例提供了一个自**定义事件的系统**来解决这个问题。父级组件可以像处理 native DOM 事件一样通过 v-on 监听子组件实例的任意事件：

<blog-post

...

v-on:enlarge-text="postFontSize += 0.1"

></blog-post>

同时子组件可以通过调用内建的 [**$emit 方法**](https://cn.vuejs.org/v2/api/#vm-emit) 并传入事件名称来触发一个事件：

<button v-on:click="$emit('enlarge-text')">

Enlarge text

</button>

有了这个 v-on:enlarge-text="postFontSize += 0.1" 监听器，父级组件就会接收该事件并更新 postFontSize 的值。

### [vm.$emit( eventName, […args] )](https://cn.vuejs.org/v2/api/#vm-emit)

* **参数**：
  + {string} eventName
  + [...args]

触发当前实例上的事件。附加参数都会传给监听器回调。

**组件上使用v-model：**

这个组件内的 <input> 必须：

* 将其 value 特性绑定到一个名叫 value 的 prop 上
* 在其 input 事件被触发时，将新的值通过自定义的 input 事件抛出

写成代码之后是这样的：

Vue.component('custom-input', {

props: ['value'],

template: `

<input

v-bind:value="value"

     v-on:input="$emit('input', $event.target.value)"

   >

`

})

现在 v-model 就应该可以在这个组件上完美地工作起来了：

<custom-input v-model="searchText"></custom-input>

**局部注册：**

在 components 选项中定义你想要使用的组件：

new Vue({

el: '#app',

components: {

'component-a': ComponentA,

'component-b': ComponentB

}

})

对于 components 对象中的每个属性来说，其属性名就是自定义元素的名字，其属性值就是这个组件的选项对象。