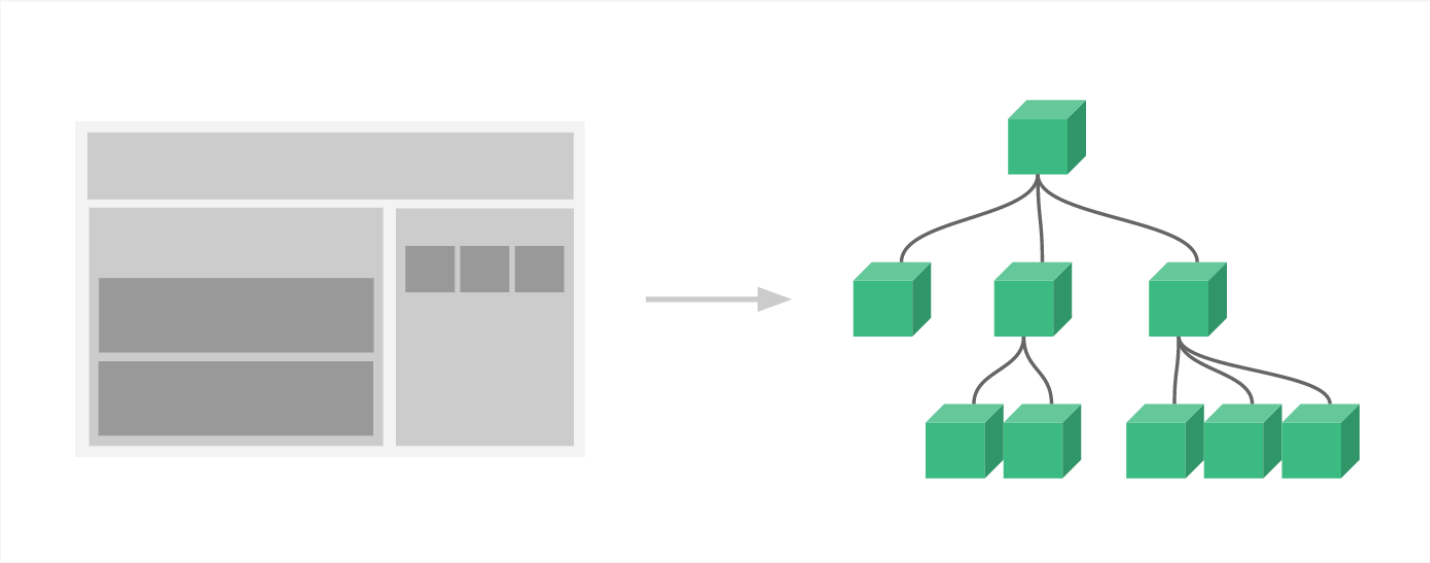
Vue.js 表单 Vue.js 自定义指令

Vue.js 组件

组件（Component）是 Vue.js 最强大的功能之一。

组件可以扩展 HTML 元素，封装可重用的代码。

组件系统让我们可以用独立可复用的小组件来构建大型应用，几乎任意类型的应用的界面都可以抽象为一个组件树：



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>
```

全局组件

所有实例都能用全局组件。

全局组件实例

注册一个简单的全局组件 runoob，并使用它：

```
<div id="app">
<runoob></runoob>
</div>
<script>
// 注册
Vue.component('runoob', {
template: '<h1>自定义组件!</h1>'
})
// 创建根实例
new Vue({
```

```
el: '#app'
})
</script>
```

尝试一下 »

局部组件

我们也可以在实例选项中注册局部组件，这样组件只能在这个实例中使用：

局部组件实例

注册一个简单的局部组件 runoob，并使用它：

```
<div id="app">
<runoob></runoob>
</div>
<script>
var Child = {
template: '<h1>自定义组件!</h1>'
}
// 创建根实例
new Vue({
el: '#app',
components: {
// <runoob> 将只在父模板可用
'runoob': Child
}
})
</script>
```

尝试一下 »

Prop

prop 是父组件用来传递数据的一个自定义属性。

父组件的数据需要通过 props 把数据传给子组件，子组件需要显式地用 props 选项声明 "prop"：

Prop 实例

```
<div id="app">
<child message="hello!"></child>
</div>
<script>
// 注册
Vue.component('child', {
// 声明 props
props: ['message'],
// 同样也可以在 vm 实例中像 "this.message" 这样使用
template: '<span>{{ message }}</span>'
})
// 创建根实例
new Vue({
el: '#app'
```

```
})
</script>
```

尝试一下 »

动态 Prop

类似于用 v-bind 绑定 HTML 特性到一个表达式，也可以用 v-bind 动态绑定 props 的值到父组件的数据中。每当父组件的数据变化时，该变化也会传导给子组件：

Prop 实例

```
<div id="app">
<div>
<input v-model="parentMsg">
<br>
<child v-bind:message="parentMsg"></child>
</div>
</div>
<script>
// 注册
Vue.component('child', {
// 声明 props
props: ['message'],
// 同样也可以在 vm 实例中像 "this.message" 这样使用
template: '<span>{{ message }}</span>'
})
// 创建根实例
new Vue({
el: '#app',
data: {
parentMsg: '父组件内容'
}
})
</script>
```

尝试一下 »

以下实例中将 v-bind 指令将 todo 传到每一个重复的组件中：

Prop 实例

```
<div id="app">
<ol>
<todo-item v-for="item in sites" v-bind:todo="item"></todo-item>
</ol>
</div>
<script>
Vue.component('todo-item', {
props: ['todo'],
template: '<li>{{ todo.text }}</li>'
})
new Vue({
el: '#app',
```

```
data: {
sites: [
{ text: 'Runoob' },
{ text: 'Google' },
{ text: 'Taobao' }
]
}
})
</script>
```

尝试一下 »

注意: prop 是单向绑定的：当父组件的属性变化时，将传导给子组件，但是不会反过来。

Prop 验证

组件可以为 props 指定验证要求。

prop 是一个对象而不是字符串数组时，它包含验证要求：

```
Vue.component('example', {
  props: {
    // 基础类型检测 （`null` 意思是任何类型都可以）
    propA: Number,
    // 多种类型
    propB: [String, Number],
    // 必传且是字符串
    propC: {
      type: String,
      required: true
    },
    // 数字，有默认值
    propD: {
      type: Number,
      default: 100
    },
    // 数组／对象的默认值应当由一个工厂函数返回
    propE: {
      type: Object,
      default: function () {
        return { message: 'hello' }
      }
    },
    // 自定义验证函数
    propF: {
      validator: function (value) {
        return value > 10
      }
    }
```

```
    }
  })
```

type 可以是下面原生构造器：

- String
- Number
- Boolean
- Function
- Object
- Array

type 也可以是一个自定义构造器，使用 instanceof 检测。

自定义事件

父组件是使用 props 传递数据给子组件，但如果子组件要把数据传递回去，就需要使用自定义事件！

我们可以使用 v-on 绑定自定义事件, 每个 Vue 实例都实现了事件接口(Events interface)，即：

- 使用 `$on(eventName)` 监听事件
- 使用 `$emit(eventName)` 触发事件

另外，父组件可以在使用子组件的地方直接用 v-on 来监听子组件触发的事件。

以下实例中子组件已经和它外部完全解耦了。它所做的只是触发一个父组件关心的内部事件。

实例

```
<div id="app">
<div id="counter-event-example">
<p>{{ total }}</p>
<button-counter v-on:increment="incrementTotal"></button-counter>
<button-counter v-on:increment="incrementTotal"></button-counter>
</div>
</div>
<script>
Vue.component('button-counter', {
template: '<button v-on:click="incrementHandler">{{ counter }}</button>',
data: function () {
return {
counter: 0
}
},
methods: {
incrementHandler: function () {
this.counter += 1
this.$emit('increment')
}
},
```

```
})
new Vue({
el: '#counter-event-example',
data: {
total: 0
},
methods: {
incrementTotal: function () {
this.total += 1
}
}
})
</script>
```

尝试一下 »

如果你想在某个组件的根元素上监听一个原生事件。可以使用 .native 修饰 v-on 。例如：

```
<my-component v-on:click.native="doTheThing"></my-component>
```

data 必须是一个函数

上面例子中，可以看到 button-counter 组件中的 data 不是一个对象，而是一个函数：

```
data: function () {
  return {
    count: 0
  }
}
```

这样的好处就是每个实例可以维护一份被返回对象的独立的拷贝，如果 data 是一个对象则会影响到其他实例，如下所示：

实例

```
<div id="components-demo3" class="demo">
<button-counter2></button-counter2>
<button-counter2></button-counter2>
<button-counter2></button-counter2>
</div>
<script>
var buttonCounter2Data = {
count: 0
}
Vue.component('button-counter2', {
/*
data: function () {
// data 选项是一个函数，组件不相互影响
return {
count: 0
}
},
```

```
*/
data: function () {
// data 选项是一个对象，会影响到其他实例
return buttonCounter2Data
},
template: '<button v-on:click="count++">点击了 {{ count }} 次。</button>'
})
new Vue({ el: '#components-demo3' })
</script>
```

尝试一下 »

← Vue.js 表单 Vue.js 自定义指令 →