vue 第五章 官网 备课[非公开]

代码仓库： GitHub

创建时间： `2019-11-25 21:35:58`

更新时间： `2019-12-1 22:05:14`

组件名

在注册一个组件的时候，我们始终需要给它一个名字。比如在全局注册的时候我们已经看到了：

```
Vue.component('my-component-name', { /* ... */ })
```

该组件名就是 `Vue.component` 的第一个参数。

你给予组件的名字可能依赖于你打算拿它来做什么。当直接在 DOM 中使用一个组件 (而不是在字符串模板或**单文件组件**) 的时候，我们强烈推荐遵循 **W3C 规范** 中的自定义组件名 (字母全小写且必须包含一个连字符)。这会帮助你避免和当前以及未来的 HTML 元素相冲突。

你可以在**风格指南** 中查阅到关于组件名的其它建议。

组件名大小写

定义组件名的方式有两种：

使用 kebab-case

```
Vue.component('my-component-name', { /* ... */ })
```

当使用 kebab-case (短横线分隔命名) 定义一个组件时，你也必须在引用这个自定义元素时使用 kebab-case，例如 `<my-component-name>` 。

使用 PascalCase

```
Vue.component('MyComponentName', { /* ... */ })
```

当使用 PascalCase (首字母大写命名) 定义一个组件时，你在引用这个自定义元素时两种命名法都可以使用。也就是说 `<my-component-name>` 和 `<MyComponentName>` 都是可接受的。注意，尽管如此，直接在 DOM (即非字符串的模板) 中使用时只有 kebab-case 是有效的。

全局注册

到目前为止，我们只用过 `Vue.component` 来创建组件：

```
Vue.component('my-component-name', {
  // ... 选项 ...
})
```

这些组件是全局注册的。也就是说它们在注册之后可以用在任何新创建的 Vue 根实例 (new Vue) 的模板中。比如：

```
Vue.component('component-a', { /* ... */ })
Vue.component('component-b', { /* ... */ })
Vue.component('component-c', { /* ... */ })

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

```
<div id="app">
  <component-a></component-a>
  <component-b></component-b>
  <component-c></component-c>
</div>
```

在所有子组件中也是如此，也就是说这三个组件在各自内部也都可以相互使用。

练习代码

```
<!DOCTYPE html>
<!--声明语言-->
<html lang="en">
    <!--头-->
    <head>
        <!-- 声明字符编码 -->
        <meta charset="UTF-8" />
        <!--  标题 显示位置  -->
        <title>vue</title>
        <style></style>
    </head>

    <!--内容-->
    <body>
        <!--视图  view-->
        <div id="hello">
            <h1>
                创建组件

                <a
                    href="https://cn.vuejs.org/v2/guide/components-
registration.html"
                    target="_blank"
                    >VUE 官网</a
                >
            </h1>

            <br />
            <br />

            <input type="text" v-model="text1" />

            <br />
            <br />
            <test1></test1>
        </div>
        <!--视图  view 模板页面 end -->
    </body>

    <!--引入vue-->
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

```
    <script type="text/javascript">
        // 定义全局指令
        Vue.component(
            "test1",
            Vue.extend({
                template: "<h2>这是自定义组件 test1 </h2>"
            })
        );

        // <!--    视图模型 vm viewmodel -->
        let vm = new Vue({
            // element 元素选择器 # id
            el: "#hello",
            // 数据对象 模型
            data: {
                text1: "fffDDeerrqeWWrtUi"
            },
            methods: {},
            // 局部写法 私有指令
            directives: {}
        });
    </script>
</html>
```

第二种写法

```
        // 定义全局指令
        Vue.component(
            "test1",
            Vue.extend({
                template: "<h2>这是自定义组件 test1 </h2>"
            })
        );

        // 定义全局指令
        Vue.component("test2", {
            template: "<h2>这是自定义组件 test2 </h2>"
        });
```

创建组件细节

```
// 思考：why
// Vue.component("test3", {
//     template:
//         "<template><h2>这是自定义组件 test3 top</h2><h3>test3
bottom<h3></template>"
// });
```

自学官网：局部注册

练习：将上面的全局组件改为局部组件

全局注册往往是不够理想的。比如，如果你使用一个像 `webpack` 这样的构建系统，全局注册所有的组件意味着即便你已经不再使用一个组件了，它仍然会被包含在你最终的构建结果中。这造成了用户下载的 JavaScript 的无谓的增加。

在这些情况下，你可以通过一个普通的 `JavaScript` 对象来定义组件：

```
var ComponentA = { /* ... */ }
var ComponentB = { /* ... */ }
var ComponentC = { /* ... */ }
```

然后在 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。

注意局部注册的组件在其子组件中不可用。例如，如果你希望 `ComponentA` 在 `ComponentB` 中可用，则你需要这样写：

```
var ComponentA = { /* ... */ }

var ComponentB = {
  components: {
    'component-a': ComponentA
  },
  // ...
}
```

或者如果你通过 Babel 和 webpack 使用 ES2015 模块，那么代码看起来更像：

```
import ComponentA from './ComponentA.vue'

export default {
  components: {
    ComponentA
  },
  // ...
}
```

注意在 ES2015+ 中，在对象中放一个类似 `ComponentA` 的变量名其实是 `ComponentA: ComponentA` 的缩写，即这个变量名同时是：

- 用在模板中的自定义元素的名称
- 包含了这个组件选项的变量名

讲解源代码[注意代码格式，空格换行]

```
<!DOCTYPE html>
<!--声明语言-->
<html lang="en">
    <!--头-->
    <head>
        <!-- 声明字符编码 -->
        <meta charset="UTF-8" />
        <!--  标题 显示位置  -->
        <title>vue</title>
        <style></style>
    </head>
```

```
    <!--内容-->
    <body>
        <!--视图  view-->
        <div id="hello">
            <h1>
                创建局部组件

                <a
                    href="https://cn.vuejs.org/v2/guide/components-
registration.html"
                    target="_blank"
                    >VUE 官网</a
                >
            </h1>

            <br />
            <br />

            <input type="text" v-model="text1" />

            <br />
            <br />
            <!-- 注意命名 只能小写 -->
            <component-a></component-a>

            <test1></test1>
            <test2></test2>
            <test3></test3>
        </div>
        <!--视图  view 模板页面 end -->
    </body>

    <!--引入vue-->
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>

    <script type="text/javascript">
        // 定义全局指令
        Vue.component(
            "test1",
            Vue.extend({
                template: "<h2>这是自定义组件 test1 </h2>"
            })
        );

        // 定义全局指令
        Vue.component("test2", {
            template: "<h2>这是自定义组件 test2 </h2>"
```

```
        });

        let ComponentA = {
            template: "<h2>这是自定义局部组件 ComponentA </h2>"
        };
        // // 或者
        // let ComponentA = Vue.extend({
        //     template: "<h2>这是自定义局部组件 ComponentA AAAAAA
</h2>"
        // });

        /*******************  注意 *******************/
        /*******************  注意 *******************/
        /*******************  注意 *******************/
        // 问题：为什么 <h3>test3 bottom<h3> 没有显示
        // 定义全局指令
        // 解决方法 最外面加个标签
        Vue.component("test3", {
            template:
                "<div><h2>这是自定义组件 test3 top</h2><h3>test3
bottom<h3></div>"
        });

        // 思考：why
        // Vue.component("test3", {
        //     template:
        //         "<template><h2>这是自定义组件 test3 top</h2>
<h3>test3 bottom<h3></template>"
        // });

        // <!--    视图模型 vm viewmodel -->
        let vm = new Vue({
            // element 元素选择器 # id
            el: "#hello",
            // 数据对象 模型
            data: {
                text1: "fffDDeerrqeWWrtUi"
            },
            components: {
                // 注意命名 或者 "component-a"
                // componentA: ComponentA
                // 可以简写为
                ComponentA
            },
            methods: {},
            // 局部写法 私有指令
            directives: {}
```

```
        });
    </script>
</html>
```

注意命名细节

Html

```
<!-- 注意命名 只能小写 -->
<component-a></component-a>
```

JavaScript

```
components: {
                // 注意命名 或者 "component-a"
                componentA: ComponentA
            },
```

简写

```
components: {
                // 注意命名 或者 "component-a"
                // componentA: ComponentA
                // 可以简写为
                ComponentA
            },
```

思考：每次定义组件时都需要把 `html` 代码作为字符串传入，有没有其他方法可以优化?

讲解：

`创建组件-新方法.html`

组件中的数据

创建时间： `2019-12-1 21:27:27`

```
// 定义全局指令
      Vue.component("custom1", {
          template: "#template1",
          data: () => {
              return {
                  val1: 100
              };
          }
      });
```

思考：

为什么和之前定义的data不一样?

```
// <!--    视图模型 vm viewmodel -->
      let vm = new Vue({
          // element 元素选择器 # id
          el: "#hello",
          // 数据对象 模型
          data: {
              text1: "fffDDeerrqeWWrtUi"
          },
          methods: {},
          // 局部写法 私有指令
          directives: {}
      });
```

再思考：

如果把 data 封装在一个对象里面，再返回这样能成功吗？

`组件中的数据-2.html`

```
        let obj = {
            val1: 200,
            val2: 300
        };
        // 定义全局指令
        Vue.component("custom1", {
            template: "#template1",
            data: () => {
                return obj;
            }
        });
```

运行：

`组件中的数据-2.html`

讲解原理：

组件切换

创建：`2019-12-1 21:38:18`

完成练习

`8-组件切换-1.html`

讲解：

`8-组件切换-2.html`

再练习：

查看官网

请自学写出组件切换，并且利用之前的知识做出切换动画。

讲解：

`8-组件切换-3.html`