Vue中组件的探索和封装

01、组件的命名

参考：https://cn.vuejs.org/v2/style-guide/#%E7%BB%84%E4%BB%B6%E5%90%8D%E4%B8%BA%E5%A4%9A%E4%B8%AA%E5%8D%95%E8%AF%8D%E5%BF%85%E8%A6%81

反例

```
Vue.component('todo', {
  // ...
})
export default {
  name: 'Todo',
  // ...
}
```

好例子

```
Vue.component('todo-item', {
  // ...
})

export default {
  name: 'TodoItem',
  // ...
}
```

总结

- 命名一定要按照驼峰命名，比如：KsButton
- 使用组件

02、组件的初始化

02-01、全局注册(使用最多)

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
    <title>21、Vue的组件的初探和注册-01.html</title>
</head>
<body>


<div id="app">
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
</div>

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

    // 1: 根组件
    var app = Vue.createApp({
        data(){

        },
        created(){

        },
        methods:{

        }
```

```
39      });
40
41      // 2: 子组件
42      app.component("KsButton", {
43          template:"<button>登录</button>",
44          data(){
45
46          },
47
48          created() {
49          },
50
51          methods:{
52
53          }
54      });
55
56      // 3:vue视图挂载
57      app.mount("#app");
58
59  </script>
60
61  </body>
62  </html>
```

02-02、局部注册

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>21、Vue的组件的初探和注册-01.html</title>
</head>
<body>


<div id="app">
    <ks-button></ks-button>
    <ks-time></ks-time>
</div>

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

    // 1: 根组件
    var app = Vue.createApp({
        data(){

        },
        created(){
```

```
27
28          },
29          methods:{
30
31          },
32          components:{
33              "KsTime":{
34                  template:"<div>我是时间</div>",
35                  data(){
36
37                  }
38              },
39              "KsButton":{
40                  template:"<button>登录</button>",
41                  data(){
42
43                  },
44
45                  created() {
46                  },
47
48                  methods:{
49
50                  }
51              }
52          }
53      }).mount("#app");
54  </script>
55
56  </body>
57  </html>
```

总结：

- 不要太纠结到底全局注册好，还是局部注册，
- 全局注册是为来开发，几乎占用大部分，比如：elementui,antd等都是全局注册
- 局部注册：局部注册的组件，只能服务与当前的Vue实例。
- 因为组件是可复用的 Vue 实例，所以它们与 `Vue.createApp({})` 接收相同的选项，例如 `data`、`computed`、`watch`、`methods` 以及生命周期钩子等。仅有的例外是像 `el` 这样根实例特有的选项。
- 特有的东西：template模板和props自定属性

03、组件的特点

- 通用和复用

  你可以将组件进行任意次数的复用：

```
<div id="app">
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
</div>

```

避免重复造轮子。

- 组件的隔离性

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>21、Vue的组件的初探和注册-01.html</title>
</head>
```

```
<body>


<div id="app">
    <ks-button></ks-button>
    <ks-button></ks-button>
    <ks-button></ks-button>
</div>



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

    // 1: 根组件
    var app = Vue.createApp({
        data(){

        },
        created(){

        },
        methods:{

        }
    });
    // 2: 子组件 -- 必须放在父组件内部
    app.component("KsButton", {
        template:"<button @click='countNum'>点我 {{count}}</button>",
        data(){
```

```
            return {
                count:1
            }
        },
        methods:{
            countNum(){
                this.count++;
            }
        }
    });



    // 3:vue视图挂载
    app.mount("#app");

</script>

</body>
</html>
```

通过上面分析得出，每个组件调用都是独立的vue实例，是不干扰和影响。

04、组件的认识是什么？

组件的来源：

- 第三方组件
  - /element-ui (美团)
  - antd （蚂蚁）
- 自定义组件
  - 很少人会自定组件

比如：现在开源项目：若依，vue-admin，antd-pro等等，都是通过依赖第三方组件开发的开源产品。比如若依：它其实就搭建各种后台的项目提供接口，在项目提供一些基础的功能，使用前后端分离的架构模型，进行接口的对接，这样可以方便企业不需要从0开始去编写后台系统，直接用基础功能，进行二次开发和迭代。服务于中小型企业会使用比较多。

总结：复用，不用重复造轮子。能够成为组件的东西：一定是通用，如果不通用不建议封装成组件。直接写在页面上即可。

04、组件自定义属性-props

组件的基本要素：

- 命名规范
- 一定非常的通用
- 组件自定属性
- 内容可以控制
- 父子组件进行通讯

05、组件自定义属性-props

参考官网：https://cn.vuejs.org/v2/guide/components-props.html

意义：自定义属性

作用：自定义属性的名字可以直接在组件的template中可以通过指令和插值表达式直接使用

命名规则：因为属性在标签上的规定都是全小写，

- 要么就全小写(建议大家用这种)
- 遵循小驼峰命名(首字母小写)
- 使用的时候大小写无所谓，因为浏览器会全部转换成小写，所以你在定义props的时候全用小写

html部分：

```
<div id="app">
    <ks-button text="登录1"></ks-button>
    <ks-button text="登录2"></ks-button>
    <ks-button text="登录3"></ks-button>
</div>
```

js部分

```

// 1: 根组件
var app = Vue.createApp({});
// 2: 子组件 -- 必须放在父组件内部
app.component("KsButton", {
```

```
6       // 自定义属性，属性的名字可以直接在组件的template中可以通过指令和插值表达式直接使用
7       props: ['text'],
8       template:"<button @click='countNum'>{{text}}</button>",
9       data(){
10          return {
11              count:1
12          }
13      },
14      methods:{
15          countNum(){
16              this.count++;
17          }
18      }
19  });
20
21
```

06、组件自定义属性-props规范定义

参考：https://cn.vuejs.org/v2/style-guide/#Prop-%E5%AE%9A%E4%B9%89%E5%BF%85%E8%A6%81

Prop 定义应该尽量详细。

在你提交的代码中，prop 的定义应该尽量详细，至少需要指定其类型。

详解

细致的 prop 定义有两个好处：

- 它们写明了组件的 API，所以很容易看懂组件的用法；
- 在开发环境下，如果向一个组件提供格式不正确的 prop，Vue 将会告警，以帮助你捕获潜在的错误来源。

反例

```
// 这样做只有开发原型系统时可以接受
props: ['status']
```

好例子

```
# 至少要用下面这种方式
props: {
  status: String
}
# 最佳 更好的做法!
props: {
  status: {
    type: String,
    required: true,
    validator: function (value) {
      return [
        'syncing',
        'synced',
```

```
        'version-conflict',
        'error'
      ].indexOf(value) !== -1
    }
  }
}
```

Props的类型参考

`type` 可以是下列原生构造函数中的一个：

- `String`
- `Number`
- `Boolean`
- `Array`
- `Object`
- `Date`
- `Function`
- `Symbol`

具体案例使用

```
<!doctype html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>21、Vue的组件的初探和注册-01.html</title>
</head>
<body>


<div id="app" Title="我是div">
    <ks-button ksdtext="2222"></ks-button>
    <ks-button ksdtext="1111"></ks-button>
    <ks-button ksdtext="2222"></ks-button>
</div>



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

    // 1: 根组件
    var app = Vue.createApp({
        data(){
            return {
                users:[1,2,3,4],
                logintext:"登录ppppp"
            }
```

```
        },
        created(){

        },
        methods:{

        }
    });
    // 2: 子组件 -- 必须放在父组件内部
    app.component("KsButton", {
        // 自定义属性，属性的名字可以直接在组件的template中可以通过指令和插值表达式直接使用
        props: {
            ksdtext:Number,
            users:Array
        },
        template:"<button @click='countNum'>{{ksdtext}}</button>",
        data(){
           return {
               count:1
           }
        },
        methods:{
            countNum(){
                this.count++;
            }
        }
    });

    // 3:vue视图挂载
```

```
60      app.mount("#app");
61
62  </script>
63
64  </body>
65  </html>
```

比如：

```
1   props: {
2       ksdtext:Number,
3       users:Array
4   },
```

上面ksdtext是一个数字，users是一个数组类型，如果在调用的过程中，传递的不是数字或者数组比如：

```
1   <ks-button ksdtext="2222" users="12332423"></ks-
    button>
```

出现警告如下：

飞哥ksdtesxt="2222" 这不是数字吗?，在标签任何属性只要用双引号或者单引号引起来的数据，都是String。只有在动态的使用v-bind属性的时候，如果你属性是数字，那么就是数字类型。

```
<div id="app" Title="我是div">
    <ks-button :ksdtext="num" :users="users"></ks-button>
    <ks-button :ksdtext="num" :users="users"></ks-button>
    <ks-button :ksdtext="num" :users="users"></ks-button>
</div>
```

上面就正确的写法。不会出现警告

总结：

告诉你一个道理，警告值是一个警醒的作用，告诉你可能你数据类型写错了，你要去纠正，并不会影响vue渲染。

Props和data和computed计算属性的优先级的问题

定义相同的属性名的时候：data > props > computed 。这里告诉你一个道理，在开发中不要去定义相同的属性名，会产生错觉。而且可能会造成不可控。这个很危险的行为。

- 无论定义在哪里，在后面在methods的行为中都获取。
- 但是props的属性属性不会渲染组件模板。
- 在定义的时候，data中属性和props的属性和computeds属性也不要和methods的方法名出现冲突，也造成影响。

父子组件直接的通讯

- props 是一种：父组件传递给子组件数据一种方式

组件的插槽**v-slot**