link: null
title: 珠峰架构师成长计划
description: https://www.w3.org/TR/html/syntax.html#void-elements
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=63 sentences=34, words=436

组件的声明

- 全局组件

```
<my-button></my-button>
Vue.component('my-button',{
  template:'<button>&#x70B9;&#x6211;&#x554A;</button>'
})
let vm = new Vue({
  el:'#app'
})
```

- 局部组件

```
<my-button></my-button>
let vm = new Vue({
  el:'#app',
  components:{
      'MyButton':{
          template:'<button>&#x6309;&#x94AE;</button>'
      }
  }
});
```

HTML不支持自闭合的自定义元素，在DOM模板里永远不要使用自闭和组件,在HTML中也不支持MyButton的写法，所以组件调用全部使用短横线连接的方式!

组件的数据

在组件中的数据必须是函数的形式

```
'MyButton':{
    data(){
        return {content:'&#x6309;&#x94AE;'}
    },
    template:'<button>{{content}}</button>'
}
```

组件的属性应用及校验

```
<my-button button-content="&#x6309;&#x94AE;"></my-button>
components:{
    'MyButton':{
        props:['buttonContent'],
        template:'<button>{{buttonContent}}</button>'
    }
}
```

属性在组件标签上需要使用短横线命名法，在组件中声明需要采用驼峰命名法

```
<my-button button-content="&#x6309;&#x94AE;" :number="'1'"></my-button>
components:{
    'MyButton':{
        props:{
            buttonContent:String,
            arr:{
                type:Array,
                default:()=>([])
            },
            number:{
                type:Number,
                validator:(value)=>{
                    return typeof value == 'number'
                }
            },

        },
        template:'<button>{{buttonContent}} {{arr}} {{number}}</button>'
    }
}
```

```
<my-button v-bind="info"></my-button>
let vm = new Vue({
    el:'#app',
    data:{
        info:{name:'&#x59DC;&#x6587;',age:18}
    },
    components:{
        'MyButton':{
            props:['name','age'],
            inheritAttrs:false,
            mounted(){
                console.log(this.$attrs)
            },
            template:'<button>{{name}} {{age}} </button>'
        }
    }
});
```

事件应用

```
<!-- 给组件绑定方法，在内部触发绑定的方法 -->
<my-button @click="change" msg="&#x6309;&#x94AE;"></my-button>
let vm = new Vue({
    el:'#app',
    methods:{
        change(){ alert('hello'); }
    },
    components:{
        'MyButton':{
            props:['msg'],
            template:`<div>
                <button @click="this.$listeners.click">{{msg}}</button>
                <button v-on="this.$listeners">{{msg}}</button>
                <button @click="$emit('click')"></button>
            </div>`
        }
    }
});
```

在组件的根元素上直接监听一个原生事件

```
<my-button @click.native="change"></my-button>
```

$parent & $child

实现收缩面板功能

```
<collapse>
    <collapse-item title="react">&#x5185;&#x5BB9;1</collapse-item>
    <collapse-item title="vue">&#x5185;&#x5BB9;2</collapse-item>
    <collapse-item title="node">&#x5185;&#x5BB9;3</collapse-item>
</collapse>
Vue.component('Collapse',{
    methods:{
        open(id){
            this.$children.forEach(child => {
                if(child._uid != id){
                    child.show = false;
                }
            });
        }
    },
    template:``
});
Vue.component('CollapseItem',{
    props:['title'],
    data(){
        return {show:true}
    },
    methods:{
        change(){
            this.$parent.open(this._uid);
            this.show =!this.show;
        }
    },
    template:`<div class="collapse-item" @click="change">
        <div class="title">{{title}}</div>

    </div>`
});
```

v-slot应用

- v-slot:可以简写成#

```
<list>
  <template v-slot:banana>&#x1F34C;</template>
  <template v-slot:apple>&#x1F34E;</template>
      &#x1F34A;
</list>
list:{
  template:`

  `
}
```

- 作用域插槽

```
<list #default="{val}">
  {{val}}
</list>
list:{
  data(){
      return {arr:[1,2,3]}
  },
  template:`
      <ul>
          <li v-for="a in arr" :key="a">
              <slot :val="a">{{a}}</slot>
          </li>
      </ul>
  `
}
```

Provide & inject

跨组件数据传递，主要为高阶插件/组件库提供用例

```
provide:{ name:'zf' },
components:{
    list:{
        inject:['name'],
        template:`<div>{{name}}</div>`
    }
}
```

父子组件数据同步

.sync 和 v-model的使用

```
<div id="app">
        {{msg}}
        <tab :msg="msg" @update:msg="change"></tab>
        <tab :msg.sync="msg"></tab>
        <tab v-model="msg"></tab>
</div>

let vm = new Vue({
    el:'#app',
    data:{
        msg:'hello'
    },
    methods:{
        change(value){
            this.msg = value
        }
    },
    components:{
        tab:{
            props:['msg'],
            methods:{
                change(){
                    this.$emit('update:msg','world')
                    this.$emit('input','world');
                }
            },
            template:`<div>
                {{msg}} <button @click="change">&#x5207;&#x6362;</button>
            </div>`
        }
    }
})
```

ref 特性

- 放在dom上表示获取当前dom元素,放到组件上表示获取当前组件实例,在v-for中获取的是集合

```
// &#x5728;&#x7EC4;&#x4EF6;&#x4E2D;&#x53EF;&#x4EE5;&#x83B7;&#x53D6;&#x7EC4;&#x4EF6;&#x5B9E;&#x4F8B;
<tab :msg="msg" @update:msg="change" ref="ele"></tab>
this.$refs.ele.change()

// &#x653E;&#x5728;dom&#x4E0A;&#x8868;&#x793A;&#x83B7;&#x53D6;&#x5F53;&#x524D;dom&#x5143;&#x7D20;

this.$refs.ele.style.border="1px solid red"

// &#x5728;v-for&#x4E2D;&#x83B7;&#x53D6;&#x7684;&#x662F;&#x96C6;&#x5408;
<template v-for="a in 3">
    <tab :msg="msg" @update:msg="change" ref="ele" :key="a"></tab>
</template>
console.log(this.$refs.ele.length)
```

- 1. props和$emit 父组件向子组件传递数据是通过prop传递的，子组件传递数据给父组件是通过$emit触发事件来做到的
- 2. $attrs和$listeners A->B->C。Vue 2.4 开始提供了$attrs和$listeners来解决这个问题
- 3. $parent,$children 智能组件木偶组件
- 4. $refs 获取实例
- 5. 父组件中通过provider来提供变量，然后在子组件中通过inject来注入变量。
- 6. envetBus 平级组件数据传递 这种情况下可以使用中央事件总线的方式
- 7. vuex状态管理

异步组件

```
Vue.component('async', function (resolve, reject) {
    setTimeout(function () {
        resolve({
            template: '<div>&#x5F02;&#x6B65;&#x7EC4;&#x4EF6;</div>'
        })
    }, 1000);
});
```

在后期我们一般配合webpack的import语法使用

递归组件

- name属性 (后期实现菜单组件)