## 1修饰符.sync和表单输入组件的语法糖原理（子组件向父组件的数据流）：

本质上是子组件改变父组件状态，vue的父子组件之间一般是通过‘prop’的父组件到子组件之间的单向数据流。但在一定的特定场景中需要子组件改变父组件的状态。

当子组件改变父组件状态时，需要在子组件用到v-on进行事件注册，并用$emit监听子组件的状态变化，同时父组件也需要用v-on来注册用$emit来监听的子组件的事件，进而在父组件注册的事件中同步子组件的状态变化。

由于v-mdel语法糖默认扩展为：

<currency-input

v-bind:value="price"

v-on:input="price=arguments[0]">

</currency-input>

所以在input用type=CheckBox时，value有可能有其他用途，此时就需要自定义v-bind和v-on要绑定的值和事件，这事就需要在子组件注册时，来自定义model，

Vue.component("my-checkbox",{

template:'\

<span>\

$\

<input\

type="checkbox"\

ref="input"\

v-bind:checked="checked"\

v-bind:value="value"\

v-on:change="updateValue($event.target.checked,$event.target.value)"\

>\

</span>\

',

model:{

prop:"checked",

event:"change"

},

props:{

checked:Boolean,

value:String

},

methods:{

}

}

})

即实际扩展为,此时的value可以用作自己想要用的值：

<currency-input

v-bind:checked="price"

v-on:change="price=arguments[0]"

v-bind:value=”some”>

</currency-input>

## 2非父子组件之间的通信

本质上是两个组件之间的事件监听（vm.$on）和事件触发（vm.$emit）,而两个组件之间是互相独立的，所以需要一个空的Vue实列（为eventBus），来作为中转站，也就是事件触发和事件监听都在空的Vue实列上进行。首先要在两个组件的生命周期mounted中监听空的Vue实列上注册的触发事件（eventBus.$on(‘自定义事件a’),回调函数里进行状态改变），然后再在各自组件上触发自定义事件（eventBus.$emit(‘自定义事件a’)）。例子：

// 注册一个空的 Vue 实例，作为 ‘中转站’

var eventBus = new Vue({})

// foo 组件

var foo = {

template: '<div><p>the count of foo is {{fooCount}}</p>' +

'<button @click="addBar">add bar\'s count</button></div>',

data: function() {

return {

fooCount: 0

}

},

methods: {

addBar: function() {

// 触发事件

eventBus.$emit('addBar')

}

},

mounted: function() {

eventBus.$on('addFoo', function(num) {

this.fooCount +=num

}.bind(this))

// 这里必须将 this 绑定在组件实例上。如果不使用 bind , 也可以使用箭头函数。

}

}

// bar 组件

var bar = {

template: '<div><p>the count of bar is {{barCount}}</p>' +

'<button @click="addFoo">add foo\'s count</button></div>',

data: function() {

return {

barCount: 0

}

},

methods: {

addFoo: function() {

// 触发事件，同时传递一个参数

eventBus.$emit('addFoo', 2)

}

},

// 在 组件创建的钩子函数中 监听事件

mounted: function() {

eventBus.$on('addBar', function() {

alert('bar')

this.barCount++

}.bind(this))

}

}

var vm = new Vue({

el: '#app8',

components: {

foo,

bar

}

})