

# 05-scoped+组件通信+props校验+记事本组件版

## 一、scoped作用及原理

### 1. scoped作用

#### 1.1 默认情况

写在组件中的样式会 **全局生效**, 因此很容易造成多个组件之间的样式冲突问题。

1. **全局样式**: 默认组件中的样式会作用到全局, 任何一个组件中都会受到此样式的影响
2. **局部样式**: 可以给组件加上**scoped** 属性, 可以让样式只作用于当前组件的标签

#### 1.2 代码示例

MyLeft.vue

```
1 <script setup></script>
2
3 <template>
4   <div
5     class="my-left"
6     style="
7       flex: 1;
8       align-items: center;
9       height: 100px;
10      background: paleturquoise;
11      text-align: center;
12    ">
13     <h3>每天起床第一句</h3>
14   </div>
15 </template>
16
17 <style>
18   h3 {
19     color: red;
20   }
21 </style>
```

## MyRight.vue

```
1 <script setup></script>
2
3 <template>
4   <div
5     class="my-right"
6     style="
7       flex: 1;
8       align-items: center;
9       height: 100px;
10      background: plum;
11      text-align: center;
12    ">
13     <h3>先给自己打个气</h3>
14   </div>
15 </template>
16
17 <style></style>
```

## App.vue

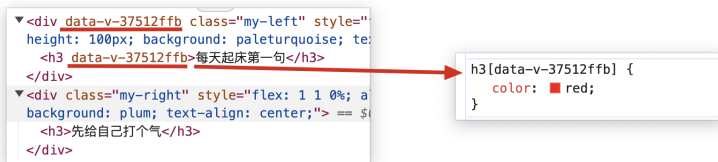
```
1 <script setup>
2   import MyLeft from './components/MyLeft.vue'
3   import MyRight from './components/MyRight.vue'
4 </script>
5 <template>
6   <div class="container">
7     <MyLeft />
8     <MyRight />
9   </div>
10 </template>
11
12 <style>
13   .container {
14     display: flex;
15   }
16 </style>
```

至此,发现 MyRight.vue 中 h3 的样式受到了的影响。为了解决样式污染/冲突问题,可以给组件的 style 添加 scoped 属性,这样就避免了不同组件之前样式污染的问题

## 2. scoped原理

1. 组件内所有标签都被添加**data-v-hash值** 的自定义属性
2. css选择器都被添加 [**data-v-hash值**] 的属性选择器

最终效果: 必须是当前组件的元素, 才会有这个自定义属性, 从而保证了样式只能作用到当前组件



### 3. 总结

1. style的默认样式是作用到哪儿的?

答: 全局

2. scoped的作用是什么?

答: 防止不同组件样式冲突(污染)

3. scoped的原理是?

答: 1、程序员书写的选择器与属性选择器[data-v-xxxx]形成交集选择器

2、因为属性选择器[data-v-xxxx]的唯一性, 保证了该样式只能对当前组件内的元素生效

4. style上推不推荐加scoped?

答: Yes、推荐

## 二、父子组件通信

### 1. 介绍

#### 1.1 什么是组件通信?

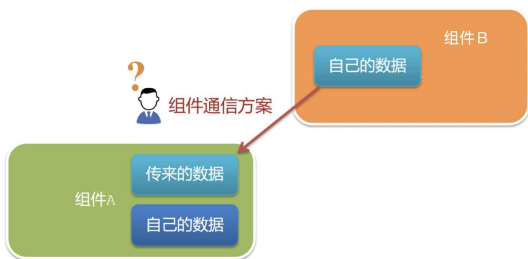
组件通信, 是指一个把数据传递给另一个组件

#### 1.2 为什么要组件通信

因为之前把代码写在一起的, 数据直接使用即可; 现在是组件化开发, 通过代码拆分和组合的方式进行开发, 这种情况下, 还要达到和不拆分之前一样的效果, 这时组件之间难免需要数据传递, 这就需要组件之间进行通信

1. 组件的数据是独立的, 无法直接访问其他组件的数据。
2. 想使用其他组件的数据, 就需要组件通信

#### 1.3 组件之间如何通信



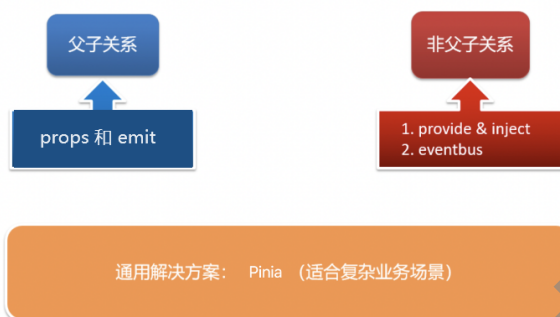
## 1.4 思考

### 1.4.1 组件之间的关系

- 1、父子关系：谁被使用, 谁就是子组件, 当前组件就是父组件
- 2、非父子关系

### 1.4.2 通信方案

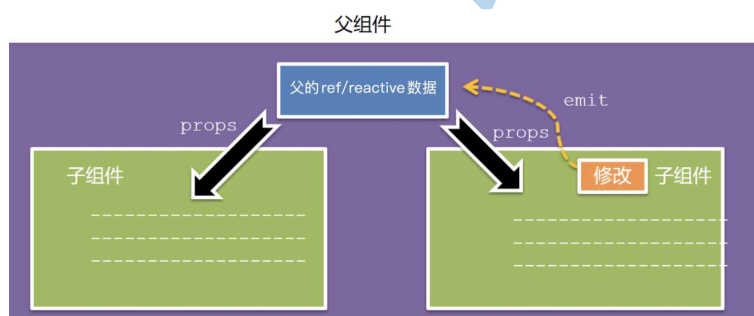
- 1、父子关系
- 2、非父子关系



## 2. 父传子

### 2.1 方法

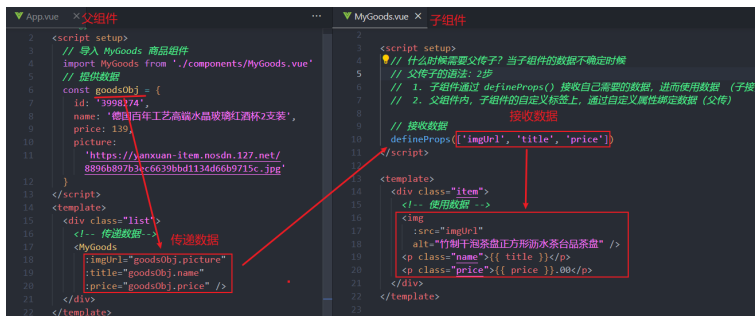
父组件通过**props**将数据传递给子组件



### 2.2 步骤

父向子传值步骤：

1. 子组件通过 `defineProps` 接收数据（子接）
2. 父组件通过 `自定义属性` 传递数据（父传）



## 2.3 静态代码(可复制)

App.vue

```
1 <script setup>
2   // 提供数据
3   const goodsObj = {
4     id: '3998274',
5     name: '德国百年工艺高端水晶玻璃红酒杯2支装',
6     price: 139,
7     picture:
8       'https://yanxuan-item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'
9   }
10 </script>
11 <template>
12   <div class="list">
13     <div class="item">
14       
18       <p class="name">竹制干泡茶盘正方形沥水茶台品茶盘</p>
19       <p class="price">139.00</p>
20     </div>
21   </div>
22 </template>
23 <style lang="scss">
24   * {
25     margin: 0;
26     padding: 0;
27     box-sizing: border-box;
28   }
29
30   .list {
31     width: 1000px;
32     margin: 0 auto;
33     display: flex;
34     flex-wrap: wrap;
```

```

35   }
36
37   .item {
38     width: 240px;
39     margin-left: 10px;
40     padding: 20px 30px;
41     transition: all 0.5s;
42     margin-bottom: 20px;
43     .item:nth-child(4n) {
44       margin-left: 0;
45     }
46
47     &:hover {
48       box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
49       transform: translate3d(0, -4px, 0);
50       cursor: pointer;
51     }
52
53     img {
54       width: 100%;
55     }
56
57     .name {
58       font-size: 18px;
59       margin-bottom: 10px;
60       color: #666;
61     }
62
63     .price {
64       font-size: 22px;
65       color: firebrick;
66     }
67
68     .price::before {
69       content: '¥';
70       font-size: 22px;
71     }
72   }
73 </style>

```

## 2.4 代码示例

子组件MyGoods.vue

```

1 <script setup>
2   defineProps(['imgUrl', 'title', 'price'])

```

```
3 </script>
4
5 <template>
6   <div class="item">
7     
10    <p class="name">{{ title}}</p>
11    <p class="price">{{ price }}.00</p>
12  </div>
13 </template>
14
15 <style lang="scss" scoped>
16   .item {
17     width: 240px;
18     margin-left: 10px;
19     padding: 20px 30px;
20     transition: all 0.5s;
21     margin-bottom: 20px;
22     .item:nth-child(4n) {
23       margin-left: 0;
24     }
25
26     &:hover {
27       box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
28       transform: translate3d(0, -4px, 0);
29       cursor: pointer;
30     }
31
32     img {
33       width: 100%;
34     }
35
36     .name {
37       font-size: 18px;
38       margin-bottom: 10px;
39       color: #666;
40     }
41
42     .price {
43       display: flex;
44       align-items: center;
45       font-size: 22px;
46       color: firebrick;
47       button {
48         margin-left: 48px;
49         font-size: 14px;
```

```
50     outline: none;
51   }
52 }
53
54 .price::before {
55   content: '¥';
56   font-size: 22px;
57 }
58 }
59 </style>
```

## 2.5 总结

### 1. 父传子的作用是？

答：为了解决子组件数据不确定的问题，从而提高子组件的灵活性

### 2. 父传子的语法是？

答：1、子接 (defineProps接)

2、父传 (自定义属性传)

## 3. 配合循环独立传值

### 3.1 目标

v-for遍历组件，每次循环独立传值

### 3.2 父组件准备数组

```
1 // 商品列表
2 const goodsList = [
3   {
4     id: '4001172',
5     name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
6     price: 289,
7     picture:
8       'https://yanxuan-
item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg'
9   },
10  {
11    id: '4001594',
12    name: '日式黑陶功夫茶组双侧把茶具礼盒装',
13    price: 288,
14    picture:
15      'https://yanxuan-
item.nosdn.127.net/3346b7b92f9563c7a7e24c7ead883f18.jpg'
```



```
16     },
17     {
18         id: '4001009',
19         name: '竹制干泡茶盘正方形沥水茶台品茶盘',
20         price: 109,
21         picture:
22             'https://yanxuan-
item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png'
23     },
24     {
25         id: '4001874',
26         name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',
27         price: 488,
28         picture:
29             'https://yanxuan-
item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png'
30     },
31     {
32         id: '4001649',
33         name: '大师监制龙泉青瓷茶叶罐',
34         price: 139,
35         picture:
36             'https://yanxuan-
item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png'
37     },
38     {
39         id: '3997185',
40         name: '与众不同的口感汝瓷白酒杯套组1壶4杯',
41         price: 108,
42         picture:
43             'https://yanxuan-
item.nosdn.127.net/8e21c794dfd3a4e8573273ddae50bce2.jpg'
44     },
45     {
46         id: '3997403',
47         name: '手工吹制更厚实白酒杯壶套装6壶6杯',
48         price: 100,
49         picture:
50             'https://yanxuan-
item.nosdn.127.net/af2371a65f60bce152a61fc22745ff3f.jpg'
51     },
52     {
53         id: '3998274',
54         name: '德国百年工艺高端水晶玻璃红酒杯2支装',
55         price: 139,
56         picture:
```

```
57     'https://yanxuan-  
    item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'  
58   }  
59 ]
```

### 3.3 代码示例

App.vue

```
1 <script setup>  
2   import MyGoods from './components/MyGoods.vue'  
3   // 商品列表  
4   const goodsList = [  
5     {  
6       id: '4001172',  
7       name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',  
8       price: 289,  
9       picture:  
10        'https://yanxuan-  
    item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg'  
11    },  
12    {  
13      id: '4001594',  
14      name: '日式黑陶功夫茶组双侧把茶具礼盒装',  
15      price: 288,  
16      picture:  
17        'https://yanxuan-  
    item.nosdn.127.net/3346b7b92f9563c7a7e24c7ead883f18.jpg'  
18    },  
19    {  
20      id: '4001009',  
21      name: '竹制干泡茶盘正方形沥水茶台品茶盘',  
22      price: 109,  
23      picture:  
24        'https://yanxuan-  
    item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png'  
25    },  
26    {  
27      id: '4001874',  
28      name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',  
29      price: 488,  
30      picture:  
31        'https://yanxuan-  
    item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png'  
32    },  
33    {
```

```
34     id: '4001649',
35     name: '大师监制龙泉青瓷茶叶罐',
36     price: 139,
37     picture:
38       'https://yanxuan-
item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png'
39   },
40   {
41     id: '3997185',
42     name: '与众不同的口感汝瓷白酒杯套组1壶4杯',
43     price: 108,
44     picture:
45       'https://yanxuan-
item.nosdn.127.net/8e21c794dfd3a4e8573273ddae50bce2.jpg'
46   },
47   {
48     id: '3997403',
49     name: '手工吹制更厚实白酒杯壶套装6壶6杯',
50     price: 100,
51     picture:
52       'https://yanxuan-
item.nosdn.127.net/af2371a65f60bce152a61fc22745ff3f.jpg'
53   },
54   {
55     id: '3998274',
56     name: '德国百年工艺高端水晶玻璃红酒杯2支装',
57     price: 139,
58     picture:
59       'https://yanxuan-
item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'
60   }
61 ]
62 </script>
63 <template>
64   <div class="list">
65     <!-- 循环遍历、独立传值-->
66     <MyGoods
67       v-for="item in goodsList"
68       :key="item.id"
69       :imgUrl="item.picture"
70       :title="item.name"
71       :price="item.price"
72     />
73   </div>
74 </template>
75
76 <style lang="scss">
```

```

77   * {
78     margin: 0;
79     padding: 0;
80     box-sizing: border-box;
81   }
82
83   .list {
84     width: 1000px;
85     margin: 0 auto;
86     display: flex;
87     flex-wrap: wrap;
88   }
89 </style>

```

## MyGoods.vue

```

1  <script setup>
2    defineProps(['imgUrl', 'title', 'price'])
3  </script>
4
5  <template>
6    <div class="item">
7      
10     <p class="name">{{ title }}</p>
11     <p class="price">
12       <span>{{ price }}</span>
13       <button>砍一刀</button>
14     </p>
15   </div>
16 </template>
17
18 <style lang="scss" scoped>
19   .item {
20     width: 240px;
21     margin-left: 10px;
22     padding: 20px 30px;
23     transition: all 0.5s;
24     margin-bottom: 20px;
25     .item:nth-child(4n) {
26       margin-left: 0;
27     }
28
29     &:hover {

```

```
30     box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
31     transform: translate3d(0, -4px, 0);
32     cursor: pointer;
33 }
34
35 img {
36     width: 100%;
37 }
38
39 .name {
40     font-size: 18px;
41     margin-bottom: 10px;
42     color: #666;
43 }
44
45 .price {
46     display: flex;
47     height: 36px;
48     align-items: center;
49     font-size: 22px;
50     color: firebrick;
51     button {
52         margin-left: 80px;
53         font-size: 14px;
54         outline: none;
55     }
56 }
57
58 .price::before {
59     content: '¥';
60     font-size: 22px;
61 }
62 }
63 </style>
```

### 3.4 总结

#### 1. 组件上能进行v-for遍历吗？

答：能、如同遍历 li 标签一样

#### 2. 遍历组件需要注意什么？

答：给组件传值

### 4. 子传父

#### 4.1 新增一个砍价功能

发现，子组件不能修改父组件传递的数据，因为 `props` 是只读的 (只可展示,不可修改);

如何让子组件修改父组件数据呢？

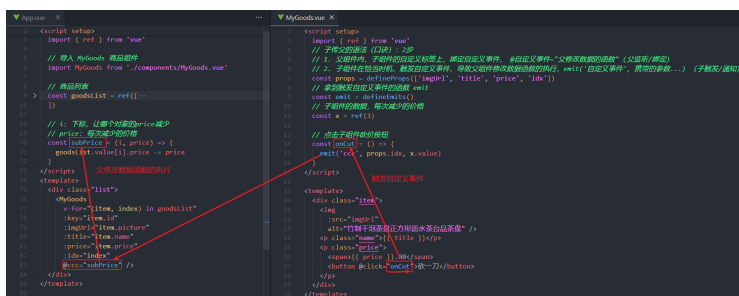
## 4.2 解决方案

子组件通知父组件，父组件收到通知后自行修改

子组件利用 **emit** 通知父组件，让组件进行修改更新

## 4.3 步骤

1. 父组件内，子组件上，绑定自定义事件，@自定义事件="父修改数据的函数"（父绑定）
2. 子组件恰当时机，触发父组件的自定义事件，`emit('自定义事件', 携带的参数...)`，从而导致父组件修改函数的时候（子触发）



## 4.4 代码示例

App.vue

```
1 <script setup>
2   import { ref } from 'vue'
3   // 导入 MyGoods 商品组件
4   import MyGoods from './components/MyGoods.vue'
5   // 商品列表
6   const goodsList = ref([
7     {
8       id: '4001172',
9       name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
10      price: 289,
11      picture:
12        'https://yanxuan-
13        item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg'
14    },
15    {
16      id: '4001594',
17      name: '日式黑陶功夫茶组双侧把茶具礼盒装',
18      price: 288,
19      picture:
```

```
19     'https://yanxuan-
item.nosdn.127.net/3346b7b92f9563c7a7e24c7ead883f18.jpg'
20 },
21 {
22     id: '4001009',
23     name: '竹制干泡茶盘正方形沥水茶台品茶盘',
24     price: 109,
25     picture:
26     'https://yanxuan-
item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png'
27 },
28 {
29     id: '4001874',
30     name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',
31     price: 488,
32     picture:
33     'https://yanxuan-
item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png'
34 },
35 {
36     id: '4001649',
37     name: '大师监制龙泉青瓷茶叶罐',
38     price: 139,
39     picture:
40     'https://yanxuan-
item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png'
41 },
42 {
43     id: '3997185',
44     name: '与众不同的口感汝瓷白酒杯套组1壶4杯',
45     price: 108,
46     picture:
47     'https://yanxuan-
item.nosdn.127.net/8e21c794dfd3a4e8573273ddae50bce2.jpg'
48 },
49 {
50     id: '3997403',
51     name: '手工吹制更厚实白酒杯壶套装6壶6杯',
52     price: 100,
53     picture:
54     'https://yanxuan-
item.nosdn.127.net/af2371a65f60bce152a61fc22745ff3f.jpg'
55 },
56 {
57     id: '3998274',
58     name: '德国百年工艺高端水晶玻璃红酒杯2支装',
59     price: 139,
```

```

60     picture:
61       'https://yanxuan-
item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'
62   }
63 ])
64
65   // i: 下标, 让哪个对象的 price 减少
66   // price: 每次减少的价格
67   const subPrice = (i, price) => {
68     goodsList.value[i].price -= price
69   }
70 </script>
71 <template>
72   <div class="list">
73     <MyGoods
74       v-for="(item, index) in goodsList"
75       :key="item.id"
76       :imgUrl="item.picture"
77       :title="item.name"
78       :price="item.price"
79       :idx="index"
80       @ccc="subPrice" />
81   </div>
82 </template>
83
84 <style lang="scss">
85   * {
86     margin: 0;
87     padding: 0;
88     box-sizing: border-box;
89   }
90
91   .list {
92     width: 1000px;
93     margin: 0 auto;
94     display: flex;
95     flex-wrap: wrap;
96   }
97 </style>

```

## MyGoods.vue

```

1 <script setup>
2   import { ref } from 'vue'
3

```



```

4 // 子传父的语法（口诀）：2步
5 // 1. 父在组件内，子组件的自定义标签上，绑定自定义事件 @自定义事件="父修改数据的函数"
  (父监听/绑定)
6 // 2. 子组件在恰当时机，通知父组件，调用 emit('自定义事件', 携带的参数...) （子触发/
  通知）
7
8 const props = defineProps(['imgUrl', 'title', 'price', 'idx'])
9
10 // 拿到触发自定义事件的函数 emit
11 const emit = defineEmits()
12
13 // 子组件的数据，表示每次减少的价格
14 const x = ref(3)
15
16 // 点击子组件砍价按钮
17 const onCut = () => {
18   // 触发 ccc 自定义事件
19   emit('ccc', props.idx, x.value)
20 }
21 </script>
22
23 <template>
24   <div class="item">
25     
28     <p class="name">{{ title }}</p>
29     <p class="price">
30       <span>{{ price }}.00</span>
31       <button @click="onCut">砍一刀</button>
32     </p>
33   </div>
34 </template>
35
36 <style scoped lang="scss">
37   .item {
38     width: 240px;
39     margin-left: 10px;
40     padding: 20px 30px;
41     transition: all 0.5s;
42     margin-bottom: 20px;
43     .item:nth-child(4n) {
44       margin-left: 0;
45     }
46
47     &:hover {
48       box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);

```

```
49     transform: translate3d(0, -4px, 0);
50     cursor: pointer;
51 }
52
53 img {
54     width: 100%;
55 }
56
57 .name {
58     font-size: 18px;
59     margin-bottom: 10px;
60     color: #666;
61 }
62
63 .price {
64     display: flex;
65     align-items: center;
66     height: 36px;
67     font-size: 22px;
68     color: firebrick;
69     button {
70         margin-left: 50px;
71         font-size: 14px;
72     }
73 }
74
75 .price::before {
76     content: '¥';
77     font-size: 22px;
78 }
79 }
80 </style>
```

## 4.5 总结

### 1. 父传子的props数据能改吗？

答：不能改，因为props是 只读的

### 2. 子传父的语法？

答：1、父绑定 -> @自定义事件="修改函数"

2、子触发 -> emit('自定义事件', 参数...)

## 三、props校验

### 1. 简易写法

## 1.1 回顾

什么是props?

就是绑定在组件标签上的自定义属性，例如：

```
1 <MyGoods
2   :imgUrl="goodsObj.picture"
3   :title="goodsObj.name"
4   :price="goodsObj.price" />
```

## 1.2 思考

组件的props可以乱传吗?

## 1.3 作用

为组件的 prop 指定 **验证要求**，不符合要求，控制台就会有**错误提示**，帮助开发者，快速发现错误

## 1.4 语法

- 类型校验
- 非空校验
- 默认值
- 自定义校验

```
const props = defineProps({
  属性名: 类型 // String/Number/Boolean/Array/Object...
})
```

## 1.5 静态代码(可复制)

MyProgress.vue

```
1 <script setup></script>
2 <template>
3   <div class="my-progress">
4     <div
5       class="inner"
6       style="width:50%">
7       <span>50%</span>
8     </div>
9   </div>
10 </template>
11
12 <style scoped>
```

```
13  .my-progress {
14    height: 26px;
15    width: 400px;
16    border-radius: 15px;
17    background-color: #272425;
18    border: 3px solid #272425;
19    box-sizing: border-box;
20    margin-bottom: 30px;
21  }
22  .inner {
23    position: relative;
24    background: #379bff;
25    border-radius: 15px;
26    height: 25px;
27    box-sizing: border-box;
28    left: -3px;
29    top: -2px;
30  }
31  .inner span {
32    position: absolute;
33    right: -30px;
34    top: 26px;
35  }
36 </style>
```

## 1.6 代码演示

App.vue

```
1 <script setup>
2   import { ref } from 'vue'
3   import MyProgress from './components/MyProgress.vue'
4   const width = ref(50)
5 </script>
6 <template>
7   <div class="app">
8     <MyProgress :width="width" />
9   </div>
10 </template>
11
12 <style></style>
```

MyProgress.vue

```

1 <script setup>
2   defineProps(['width'])
3 </script>
4 <template>
5   <div class="my-progress">
6     <div
7       class="inner"
8       :style="{ width: width + '%' }">
9       <span>{{ width }}%</span>
10    </div>
11  </div>
12 </template>
13
14 <style scoped>
15   .my-progress {
16     height: 26px;
17     width: 400px;
18     border-radius: 15px;
19     background-color: #272425;
20     border: 3px solid #272425;
21     box-sizing: border-box;
22     margin-bottom: 30px;
23   }
24   .inner {
25     position: relative;
26     background: #379bff;
27     border-radius: 15px;
28     height: 25px;
29     box-sizing: border-box;
30     left: -3px;
31     top: -2px;
32   }
33   .inner span {
34     position: absolute;
35     right: -30px;
36     top: 26px;
37   }
38 </style>

```

## 2. 完整写法

### 2.1 语法

```

1 defineProps({
2   属性名: {

```

```

3   type: 类型, // Number String Boolean ...
4   required: true, // 是否必填
5   default: 默认值, // 默认值
6
7   // value: 父传子的值
8   validator (value) {
9     // 自定义校验逻辑
10    return 布尔值
11  }
12 }
13 })

```

## 2.2 代码示例

```

1 <script setup>
2   defineProps({
3     width: {
4       type: Number,
5       // required: true
6       default: 0,
7       validator: (value) => {
8         if (value >= 0 && value <= 100) {
9           return true
10        } else {
11          console.error('The width prop must be between 0 and 100')
12          return false
13        }
14      }
15    })
16 </script>

```

## 2.3 注意

1. default和required 不同时写（因为当是必填项时，默认值也就无效了）
2. default后面如果是简单类型的值，可以直接写默认。如果是复杂类型的值，则需要以函数的形式 return一个默认值

## 2.4 总结

1. props校验的完整写法是？

答: type、required、default、validator(value)

## 3. 组件的 ref/reactive 与props 的区别

### 3.1 共同点

都可以给组件提供数据

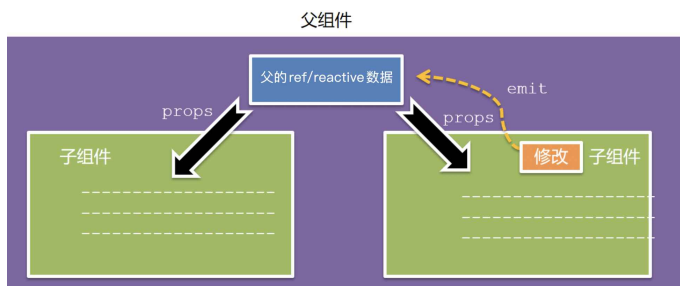
### 3.2 区别

- ref/reactive 的数据是**自己的** → 随便改
- prop 的数据是**外部的** → 不能直接改，要遵循 **单向数据流**

### 3.3 单向数据流

父级props 的数据更新，会向下流动，影响子组件。这个数据流动是单向的

### 3.4 props图示



### 3.5 口诀

谁的数据谁负责

### 3.6 总结

#### 1. 组件ref/reactive 与props 的区别？

答: 1、相同点: 都可以为组件提供数据(进行使用)

2、不同点: ref/reactive数据是组件自己的(可读可写), props数据是父给的(只读的)

## 四、记事本组件版

### 1. 整体拆分与准备

#### 1.1 组件拆分

拆分成 3 个组件, TodoHeader、TodoMain、TodoFooter, 如下图所示



## 1.2 静态代码

components/ToDoHeader.vue

```
1 <script setup></script>
2
3 <template>
4   <header class="header">
5     <h1>比特人的记事本</h1>
6     <input class="new-todo" placeholder="What needs to be finished?" autofocus
7   />
8 </header>
9 </template>
```

components/ToDoMain.vue

```
1 <script setup></script>
2
3 <template>
4   <section class="main">
5     <input id="toggle-all" class="toggle-all" type="checkbox" />
6     <label for="toggle-all">Mark all as complete</label>
7     <ul class="todo-list">
8       <li>
9         <div class="view">
10           <input class="toggle" type="checkbox" />
11           <label>学Vue组件通信</label>
12           <button class="destroy"></button>
13         </div>
14       </li>
15       <li>
16         <div class="view">
17           <input class="toggle" type="checkbox" checked />
18           <label>打王者</label>
19           <button class="destroy"></button>
20         </div>
21       </li>
22     </ul>
23   </section>
24 </template>
```

components/ToDoFooter.vue



```
1 <script setup></script>
2
3 <template>
4   <footer class="footer">
5     <span class="todo-count"><strong>0</strong> item left</span>
6     <ul class="filters">
7       <li>
8         <a href="#" class="selected">All</a>
9       </li>
10      <li>
11        <a href="#/active">Active</a>
12      </li>
13      <li>
14        <a href="#/completed">Completed</a>
15      </li>
16    </ul>
17    <button class="clear-completed">
18      Clear completed
19    </button>
20  </footer>
21 </template>
```

assets/style.css

```
1 @charset 'utf-8';
2
3 html,
4 body {
5   margin: 0;
6   padding: 0;
7 }
8
9 button {
10   margin: 0;
11   padding: 0;
12   border: 0;
13   background: none;
14   font-size: 100%;
15   vertical-align: baseline;
16   font-family: inherit;
17   font-weight: inherit;
18   color: inherit;
19   -webkit-appearance: none;
20   appearance: none;
21   -webkit-font-smoothing: antialiased;
```

```
22 -moz-osx-font-smoothing: grayscale;
23 }
24
25 body {
26   font:
27     14px 'Helvetica Neue',
28     Helvetica,
29     Arial,
30     sans-serif;
31   line-height: 1.4em;
32   background: #f5f5f5;
33   color: #111111;
34   min-width: 230px;
35   max-width: 550px;
36   margin: 0 auto;
37   -webkit-font-smoothing: antialiased;
38   -moz-osx-font-smoothing: grayscale;
39   font-weight: 300;
40 }
41
42 .hidden {
43   display: none;
44 }
45
46 .todoapp {
47   background: #fff;
48   margin: 130px 0 40px 0;
49   position: relative;
50   box-shadow:
51     0 2px 4px 0 rgba(0, 0, 0, 0.2),
52     0 25px 50px 0 rgba(0, 0, 0, 0.1);
53 }
54
55 .todoapp input::-webkit-input-placeholder {
56   font-style: italic;
57   font-weight: 400;
58   color: rgba(0, 0, 0, 0.4);
59 }
60
61 .todoapp input::-moz-placeholder {
62   font-style: italic;
63   font-weight: 400;
64   color: rgba(0, 0, 0, 0.4);
65 }
66
67 .todoapp input::input-placeholder {
68   font-style: italic;
```

```
69 font-weight: 400;
70 color: rgba(0, 0, 0, 0.4);
71 }
72
73 .todoapp h1 {
74   position: absolute;
75   top: -100px;
76   width: 100%;
77   font-size: 50px;
78   font-weight: 200;
79   text-align: center;
80   color: #b83f45;
81   -webkit-text-rendering: optimizeLegibility;
82   -moz-text-rendering: optimizeLegibility;
83   text-rendering: optimizeLegibility;
84 }
85
86 .new-todo,
87 .edit {
88   position: relative;
89   margin: 0;
90   width: 100%;
91   font-size: 24px;
92   font-family: inherit;
93   font-weight: inherit;
94   line-height: 1.4em;
95   color: inherit;
96   padding: 6px;
97   border: 1px solid #999;
98   box-shadow: inset 0 -1px 5px 0 rgba(0, 0, 0, 0.2);
99   box-sizing: border-box;
100   -webkit-font-smoothing: antialiased;
101   -moz-osx-font-smoothing: grayscale;
102 }
103
104 .new-todo {
105   padding: 16px 16px 16px 60px;
106   height: 65px;
107   border: none;
108   background: rgba(0, 0, 0, 0.003);
109   box-shadow: inset 0 -2px 1px rgba(0, 0, 0, 0.03);
110 }
111
112 .main {
113   position: relative;
114   z-index: 2;
115   border-top: 1px solid #e6e6e6;
```

```
116 }
117
118 .toggle-all {
119   width: 1px;
120   height: 1px;
121   border: none; /* Mobile Safari */
122   opacity: 0;
123   position: absolute;
124   right: 100%;
125   bottom: 100%;
126 }
127
128 .toggle-all + label {
129   display: flex;
130   align-items: center;
131   justify-content: center;
132   width: 45px;
133   height: 65px;
134   font-size: 0;
135   position: absolute;
136   top: -65px;
137   left: -0;
138 }
139
140 .toggle-all + label:before {
141   content: ' ';
142   display: inline-block;
143   font-size: 22px;
144   color: #949494;
145   padding: 10px 27px 10px 27px;
146   -webkit-transform: rotate(90deg);
147   transform: rotate(90deg);
148 }
149
150 .toggle-all:checked + label:before {
151   color: #484848;
152 }
153
154 .todo-list {
155   margin: 0;
156   padding: 0;
157   list-style: none;
158 }
159
160 .todo-list li {
161   position: relative;
162   font-size: 24px;
```

```

163 border-bottom: 1px solid #ededed;
164 }
165
166 .todo-list li:last-child {
167     border-bottom: none;
168 }
169
170 .todo-list li.editing {
171     border-bottom: none;
172     padding: 0;
173 }
174
175 .todo-list li.editing .edit {
176     display: block;
177     width: calc(100% - 43px);
178     padding: 12px 16px;
179     margin: 0 0 0 43px;
180 }
181
182 .todo-list li.editing .view {
183     display: none;
184 }
185
186 .todo-list li .toggle {
187     text-align: center;
188     width: 40px;
189     /* auto, since non-WebKit browsers doesn't support input styling */
190     height: auto;
191     position: absolute;
192     top: 0;
193     bottom: 0;
194     margin: auto 0;
195     border: none; /* Mobile Safari */
196     -webkit-appearance: none;
197     appearance: none;
198 }
199
200 .todo-list li .toggle {
201     opacity: 0;
202 }
203
204 .todo-list li .toggle + label {
205     /*
206      Firefox requires `#` to be escaped -
207      https://bugzilla.mozilla.org/show_bug.cgi?id=922433
208      IE and Edge requires *everything* to be escaped to render, so we do that
209      instead of just the `#` - https://developer.microsoft.com/en-us/microsoft-

```

[edge/platform/issues/7157459/](https://edge/platform/issues/7157459/)

```
208 */
209 background-image:
url('data:image/svg+xml;utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%20%20width%3D%2240%22%20height%3D%2240%22%20viewBox%3D%22-10%20-18%20100%20135%22%3E%3Ccircle%20cx%3D%2250%22%20cy%3D%2250%22%20r%3D%2250%22%20fill%3D%22none%22%20stroke%3D%22%23949494%22%20stroke-width%3D%223%22%2F%3E%3C%2Fsvg%3E')');
210 background-repeat: no-repeat;
211 background-position: center left;
212 }
213
214 .todo-list li .toggle:checked + label {
215   background-image:
url('data:image/svg+xml;utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20width%3D%2240%22%20height%3D%2240%22%20viewBox%3D%22-10%20-18%20100%20135%22%3E%3Ccircle%20cx%3D%2250%22%20cy%3D%2250%22%20r%3D%2250%22%20fill%3D%22none%22%20stroke%3D%22%2359A193%22%20stroke-width%3D%223%22%2F%3E%3Cpath%20fill%3D%22%233EA390%22%20d%3D%22M72%2025L42%2071%2027%2056l-4%204%2020%2020%2034-52z%22%2F%3E%3C%2Fsvg%3E')');
216 }
217
218 .todo-list li label {
219   overflow-wrap: break-word;
220   padding: 15px 15px 15px 60px;
221   display: block;
222   line-height: 1.2;
223   transition: color 0.4s;
224   font-weight: 400;
225   color: #484848;
226 }
227
228 .todo-list li.completed label {
229   color: #949494;
230   text-decoration: line-through;
231 }
232
233 .todo-list li .destroy {
234   display: none;
235   position: absolute;
236   top: 0;
237   right: 10px;
238   bottom: 0;
239   width: 40px;
240   height: 40px;
241   margin: auto 0;
242   font-size: 30px;
```

```
243     color: #949494;
244     transition: color 0.2s ease-out;
245 }
246
247 .todo-list li .destroy:hover,
248 .todo-list li .destroy:focus {
249     color: #c18585;
250 }
251
252 .todo-list li .destroy:after {
253     content: 'x';
254     display: block;
255     height: 100%;
256     line-height: 1.1;
257 }
258
259 .todo-list li:hover .destroy {
260     display: block;
261 }
262
263 .todo-list li .edit {
264     display: none;
265 }
266
267 .todo-list li.editing:last-child {
268     margin-bottom: -1px;
269 }
270
271 .footer {
272     padding: 10px 15px;
273     height: 20px;
274     text-align: center;
275     font-size: 15px;
276     border-top: 1px solid #e6e6e6;
277 }
278
279 .footer:before {
280     content: '';
281     position: absolute;
282     right: 0;
283     bottom: 0;
284     left: 0;
285     height: 50px;
286     overflow: hidden;
287     box-shadow:
288         0 1px 1px rgba(0, 0, 0, 0.2),
289         0 8px 0 -3px #f6f6f6,
```

```
290     0 9px 1px -3px rgba(0, 0, 0, 0.2),
291     0 16px 0 -6px #f6f6f6,
292     0 17px 2px -6px rgba(0, 0, 0, 0.2);
293 }
294
295 .todo-count {
296     float: left;
297     text-align: left;
298 }
299
300 .todo-count strong {
301     font-weight: 300;
302 }
303
304 .filters {
305     margin: 0;
306     padding: 0;
307     list-style: none;
308     position: absolute;
309     right: 0;
310     left: 0;
311 }
312
313 .filters li {
314     display: inline;
315 }
316
317 .filters li a {
318     color: inherit;
319     margin: 3px;
320     padding: 3px 7px;
321     text-decoration: none;
322     border: 1px solid transparent;
323     border-radius: 3px;
324 }
325
326 .filters li a:hover {
327     border-color: #db7676;
328 }
329
330 .filters li a.selected {
331     border-color: #ce4646;
332 }
333
334 .clear-completed,
335 html .clear-completed:active {
336     float: right;
```



```
337 position: relative;
338 line-height: 19px;
339 text-decoration: none;
340 cursor: pointer;
341 }
342
343 .clear-completed:hover {
344     text-decoration: underline;
345 }
346
347 .info {
348     margin: 65px auto 0;
349     color: #4d4d4d;
350     font-size: 11px;
351     text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);
352     text-align: center;
353 }
354
355 .info p {
356     line-height: 1;
357 }
358
359 .info a {
360     color: inherit;
361     text-decoration: none;
362     font-weight: 400;
363 }
364
365 .info a:hover {
366     text-decoration: underline;
367 }
368
369 /*
370     Hack to remove background from Mobile Safari.
371     Can't use it globally since it destroys checkboxes in Firefox
372 */
373 @media screen and (-webkit-min-device-pixel-ratio: 0) {
374     .toggle-all,
375     .todo-list li .toggle {
376         background: none;
377     }
378
379     .todo-list li .toggle {
380         height: 40px;
381     }
382 }
383
```

```

384 @media (max-width: 430px) {
385   .footer {
386     height: 50px;
387   }
388
389   .filters {
390     bottom: 10px;
391   }
392 }
393
394 :focus,
395 .toggle:focus + label,
396 .toggle-all:focus + label {
397   box-shadow: 0 0 2px 2px #cf7d7d;
398   outline: 0;
399 }

```

## App.vue

```

1 <script setup>
2 import { ref } from 'vue'
3 import './assets/style.css'
4 import TodoHeader from './components/TodoHeader.vue'
5 import TodoMain from './components/TodoMain.vue'
6 import TodoFooter from './components/TodoFooter.vue'
7
8 const todoList = ref([
9   { id: 321, name: '学Vue组件通信', finished: false },
10  { id: 127, name: '打王者', finished: true },
11  { id: 666, name: '跑步1小时', finished: false }
12 ])
13 </script>
14
15 <template>
16   <section class="todoapp">
17     <todo-header />
18     <todo-main />
19     <todo-footer />
20   </section>
21 </template>

```

## 2. 列表渲染

### 2.1 思路分析

- 提供数据：提供在公共的父组件 App.vue
- 通过父传子，将数据传递给TodoMain
- 利用v-for进行渲染

## 2.2 完整代码

TodoMain.vue

```
1 <script setup>
2 const props = defineProps({
3   // 数据列表
4   todoList: {
5     type: Array,
6     default: () => []
7   }
8 })
9 </script>
10
11 <template>
12   <section class="main">
13     <input id="toggle-all" class="toggle-all" type="checkbox" />
14     <label for="toggle-all">Mark all as complete</label>
15     <ul class="todo-list">
16       <li v-for="item in props.todoList"
17         :key="item.id"
18         :class="{ completed: item.finished }">
19         <div class="view">
20           <input class="toggle"
21             type="checkbox"
22             v-model="item.finished" />
23           <label>{{ item.name }}</label>
24           <button class="destroy"></button>
25         </div>
26       </li>
27     </ul>
28   </section>
29 </template>
```

App.vue

```
1 <todo-main :todoList="todoList" />
```

## 3. 添加功能

### 3.1 思路分析

1. 收集表单数据 v-model
2. 监听时间（回车+点击 都要进行添加）
3. 子传父，将任务名称传递给父组件App.vue
4. 父组件接受到数据后 进行添加 **unshift**(自己的数据自己负责)

### 3.2 完整代码

TodoHeader.vue

```
1 <script setup>
2 import { ref } from 'vue'
3
4 const emit = defineEmits()
5
6 // 任务名称
7 const title = ref('')
8
9 // 回车
10 const onEnter = () => {
11   // 非空校验
12   if (!title.value) return alert('任务名称不能为空')
13   // 通知父组件
14   emit('add-todo', title.value)
15   // 清空输入框
16   title.value = ''
17 }
18 </script>
19
20 <template>
21   <header class="header">
22     <h1>比特人的记事本</h1>
23     <input
24       class="new-todo"
25       placeholder="What needs to be finished?"
26       autofocus
27       v-model.trim="title"
28       @keyup.enter="onEnter"
29     />
30   </header>
31 </template>
32
```

## App.vue

```
1 <script setup>
2 // ...
3
4 // 添加
5 const addTodo = (name) => {
6   // 添加至数组头部
7   todoList.value.unshift({
8     id: Date.now(), // 时间戳
9     name,
10    finished: false
11  })
12 }
13 </script>
14
15 <template>
16   <section class="todoapp">
17     <todo-header @add-todo="addTodo" />
18   </section>
19 </template>
20
```

## 4. 删除功能

### 4.1 思路分析

- 监听删除的点击并携带下标
- 子传父, 将删除的下标传递给父组件App.vue
- 进行 splice 删除 (自己的数据自己负责)

### 4.2 完整代码

## TodoMain.vue

```
1 <script setup>
2   const emit = defineEmits()
3 </script>
4
5 <template>
6   <section class="main">
7     <ul class="todo-list">
```

```

8      <li
9        v-for="(item, index) in props.todoList">
10      <div class="view">
11        <button @click="emit('del-todo', index)"></button>
12      </div>
13    </li>
14  </ul>
15 </section>
16 </template>

```

App.vue

```

1 <script setup>
2
3 // 删除
4 const delTodo = (i) => {
5   if (window.confirm('确认删除么?')) {
6     todoList.value.splice(i, 1)
7   }
8 }
9 </script>
10
11 <template>
12   <todo-main @del-todo="delTodo" />
13 </template>

```

## 5. 底部功能及持久化存储

### 5.1 思路分析

1. 底部合计：父组件传递 todoList 到底部组件 —> 展示合计
2. 清除已完成功能：监听事件 —> **子组件**通知父组件 —> 父组件 filter 清除
3. 持久化存储：watch 监听数据变化，持久化到本地

### 5.2 完整代码

TodoFooter.vue

```

1 <script setup>
2 const props = defineProps({
3   todoList: {
4     type: Array,
5     default: () => []

```

```

6   }
7 })
8 const emit = defineEmits()
9 </script>
10
11 <template>
12   <footer class="footer">
13     <span class="todo-count"
14       ><strong>{{ props.todoList.length }}</strong> item left</span>
15     >
16     <button class="clear-completed" @click="emit('clear-todo')">Clear
17     completed</button>
18   </footer>
19 </template>

```

### App.vue

```

1 <script setup>
2 import { watch } from 'vue'
3 // 本地存储键名
4 const TODO_LIST_KEY = 'todo-list-key'
5 const todoList = ref(
6   // 取；有则用、否则用默认值
7   JSON.parse(localStorage.getItem(TODO_LIST_KEY)) || [
8     { id: 321, name: '学Vue组件通信', finished: false },
9     { id: 127, name: '打王者', finished: true },
10    { id: 666, name: '跑步1小时', finished: false }
11  ]
12 )
13
14 // 清除已完成
15 const clearTodo = () => {
16   todoList.value = todoList.value.filter((todo) => !todo.finished)
17 }
18
19 // 监视 todoList 的变化
20 watch(
21   todoList,
22   (newVal) => {
23     // 存
24     localStorage.setItem(TODO_LIST_KEY, JSON.stringify(newVal))
25   },
26   { // 深度监视
27     deep: true
28   }

```

```
29 )
30 </script>
31
32 <template>
33   <todo-footer :todoList="todoList" @clear-todo="clearTodo" />
34 </template>
```

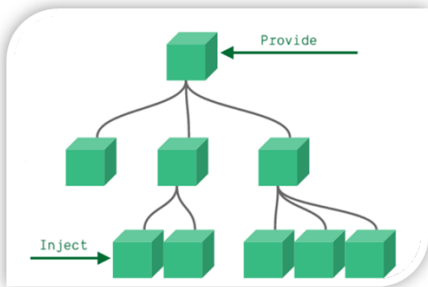
## 五、非父子组件通信

### 1. 祖先传后代 - provide & inject

#### 1.1 作用

跨层级共享数据

#### 1.2 场景



#### 1.3 语法

##### 1. 祖先组件通过 **provide** 给后代提供数据

```
1 import { provide } from 'vue'
2 provide('数据名称', 值)
```

##### 2. 后代(子/孙/曾孙...)组件通过 **inject** 获取数据

```
1 import { inject } from 'vue'
2 const 值 = inject('数据名称')
```

#### 1.4 总结

##### 1. 祖先组件如何给后代组件传值？

答：1、祖先组件: `provide('数据名称', value)`

2、后代组件: `const value = inject('数据名称')`

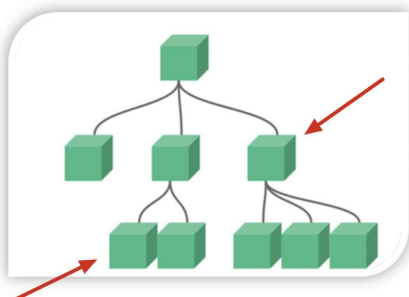


## 2. 任意两个组件通信 - EventBus

### 2.1 作用

任意两个组件通信

### 2.2 场景



### 2.3 语法

1. 下载 `mitt` 模块包，并创建公共中间人模块(类似媒婆和中介)

```
1 yarn add mitt -S
```

```
1 import mitt from 'mitt'
2 const meipo = mitt()
```

1. 确实消息发送方组件

2. `meipo.emit('消息名称', 数据...)`

3. `MyLeft.vue`

```
1 <script setup>
2
3 </script>
4
5 <template>
6   <div
7     class="my-left"
8     style="
9       flex: 1;
10      align-items: center;
11      height: 100px;
12      background: plum;

```

```

13     text-align: center;
14     ">
15     <h3>我是牛郎， 点击发送消息给织女</h3>
16     <button>Send Message</button>
17   </div>
18 </template>
19
20 <style></style>

```

### 1. 确定消息接收方组件

2. `meipo.on('消息名称', (...args) => { })`

### 3. MyRight.vue

```

1 <script setup>
2
3 </script>
4
5 <template>
6   <div
7     class="my-right"
8     style="
9       flex: 1;
10      align-items: center;
11      height: 100px;
12      background: paleturquoise;
13      text-align: center;
14    ">
15     <h3>我是织女， 收到来自牛郎的消息： xxx </h3>
16   </div>
17 </template>
18
19 <style></style>

```

### 1. App.vue

```

1 <script setup>
2   import MyLeft from './components/MyLeft.vue'
3   import MyRight from './components/MyRight.vue'
4 </script>
5 <template>
6   <div class="container">
7     <MyLeft />

```

```
8     <MyRight />
9   </div>
10 </template>
11
12 <style>
13   .container {
14     display: flex;
15   }
16 </style>
```

## 2.4 总结

### 1. 非父子关系的任意两个组件如何传值？

答: 1、中间人

2、发送方 - `emit()`

3、接收方 - `on()`