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

```
<script setup></script>

<template>
  <div
    class="my-left"
    style="
      flex: 1;
      align-items: center;
      height: 100px;
      background: paleturquoise;
      text-align: center;
    ">
    <h3>每天起床第一句</h3>
  </div>
</template>

<style>
  h3 {
    color: red;
  }
</style>
```

MyRight.vue

```
<script setup></script>

<template>
  <div
    class="my-right"
    style="
      flex: 1;
      align-items: center;
      height: 100px;
      background: plum;
      text-align: center;
    ">
    <h3>先给自己打个气</h3>
  </div>
</template>

<style></style>
```

App.vue

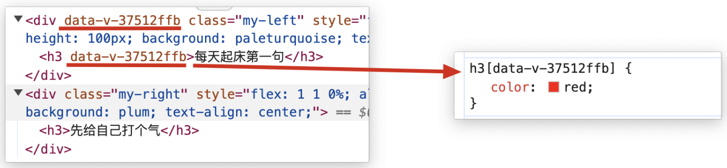
```
<script setup>
  import MyLeft from './components/MyLeft.vue'
  import MyRight from './components/MyRight.vue'
</script>
<template>
  <div class="container">
    <MyLeft />
    <MyRight />
  </div>
</template>

<style>
  .container {
    display: flex;
  }
</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



```
▼<div data-v-37512ffb class="my-left" style="
  height: 100px; background: paleturquoise; te
    <h3 data-v-37512ffb>每天起床第一句</h3>
  </div>
▼<div class="my-right" style="flex: 1 1 0%; a
  background: plum; text-align: center;"> == $
    <h3>先给自己打个气</h3>
  </div>
```

```
h3[data-v-37512ffb] {
    color: 🟥 red;
}
```

3. 总结

1. style的默认样式是作用到哪里的?

   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?

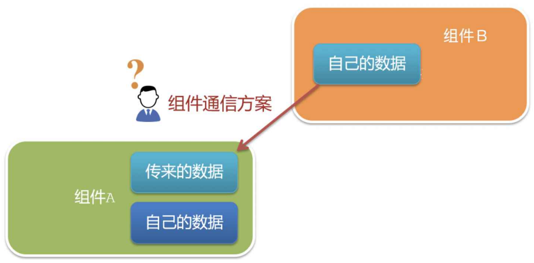
组件通信，是指一个把数组传递给另一个组件

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

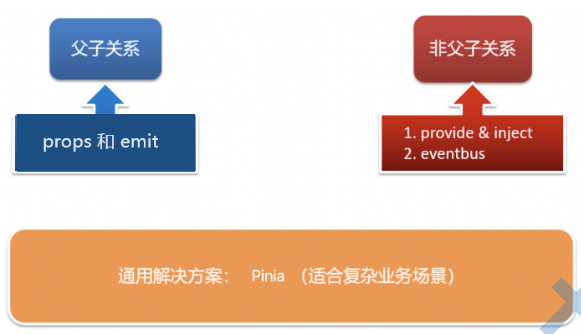
1、父子关系：谁被使用，谁就是子组件，当前组件就是父组件

2、非父子关系

1.4.2 通信方案

1、父子关系

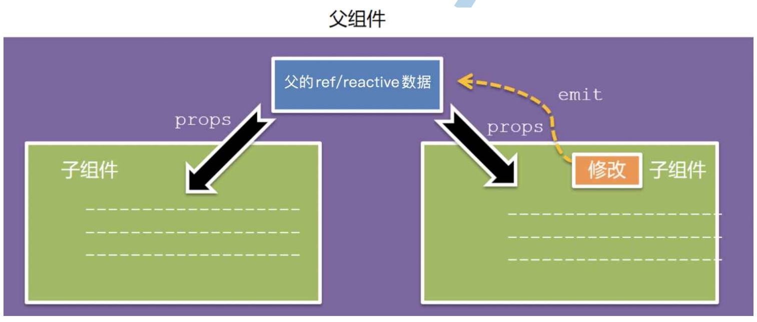
2、非父子关系



2. 父传子

2.1 方法

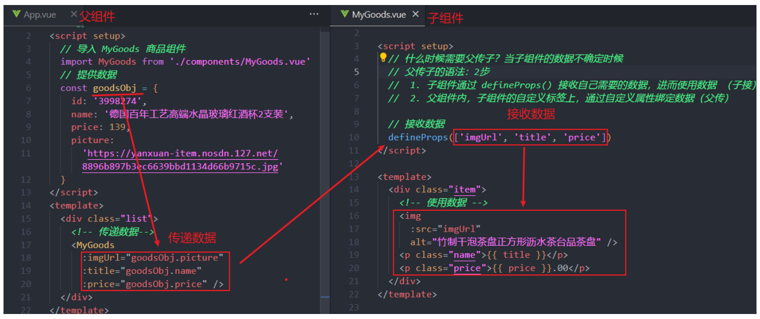
父组件通过**props**将数据传递给子组件



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）



```
<script setup>
  // 导入 MyGoods 商品组件
  import MyGoods from './components/MyGoods.vue'
  // 提供数据
  const goodsObj = {
    id: '3998274',
    name: '德国百年工艺高端水晶玻璃红酒杯2支装',
    price: 139,
    picture:
      'https://yanxuan-item.nosdn.127.net/
8896b897b3ec6639bbd1134d66b9715c.jpg'
  }
</script>
<template>
  <div class="list">
    <!-- 传递数据-->
    <MyGoods
      :imgUrl="goodsObj.picture"
      :title="goodsObj.name"
      :price="goodsObj.price" />
  </div>
</template>
```

```

<script setup>
// 什么时候需要父传子? 当子组件的数据不确定时候
// 父传子的语法: 2步
//  1. 子组件通过 defineProps() 接收自己需要的数据, 进而使用数据 (子接)
//  2. 父组件内, 子组件的自定义标签上, 通过自定义属性绑定数据 (父传)

// 接收数据
defineProps(['imgUrl', 'title', 'price'])
</script>

<template>
  <div class="item">
    <!-- 使用数据 -->
    <img
      :src="imgUrl"
      alt="竹制干泡茶盘正方形沥水茶台品茶盘" />
    <p class="name">{{ title }}</p>
    <p class="price">{{ price }}.00</p>
  </div>
</template>
```

2.3 静态代码(可复制)

App.vue

```
<script setup>
  // 提供数据
  const goodsObj = {
    id: '3998274',
    name: '德国百年工艺高端水晶玻璃红酒杯2支装',
    price: 139,
    picture:
      'https://yanxuan-item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'
  }
</script>
<template>
  <div class="list">
    <div class="item">
      <img
        src="https://yanxuan-
item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png"
        alt="竹制干泡茶盘正方形沥水茶台品茶盘" />
      <p class="name">竹制干泡茶盘正方形沥水茶台品茶盘</p>
      <p class="price">139.00</p>
    </div>
  </div>
</template>

<style lang="scss">
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  .list {
    width: 1000px;
    margin: 0 auto;
    display: flex;
    flex-wrap: wrap;
  }

  .item {
    width: 240px;
    margin-left: 10px;
    padding: 20px 30px;
    transition: all 0.5s;
    margin-bottom: 20px;
    .item:nth-child(4n) {
      margin-left: 0;
    }

    &:hover {
      box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
      transform: translate3d(0, -4px, 0);
      cursor: pointer;
    }

    img {
      width: 100%;
    }

    .name {
      font-size: 18px;
      margin-bottom: 10px;
      color: #666;
    }

    .price {
      font-size: 22px;
      color: firebrick;
    }

    .price::before {
      content: '¥';
      font-size: 22px;
    }
  }
</style>
```

2.4 代码示例

子组件MyGoods.vue

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

```
</script>

<template>
  <div class="item">
    <img
      :src="imgUrl"
      :alt="title" />
    <p class="name">{{ title}}</p>
    <p class="price">{{ price }}.00</p>
  </div>
</template>

<style lang="scss" scoped>
  .item {
    width: 240px;
    margin-left: 10px;
    padding: 20px 30px;
    transition: all 0.5s;
    margin-bottom: 20px;
    .item:nth-child(4n) {
      margin-left: 0;
    }

    &:hover {
      box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
      transform: translate3d(0, -4px, 0);
      cursor: pointer;
    }

    img {
      width: 100%;
    }

    .name {
      font-size: 18px;
      margin-bottom: 10px;
      color: #666;
    }

    .price {
      display: flex;
      align-items: center;
      font-size: 22px;
      color: firebrick;
      button {
        margin-left: 48px;
        font-size: 14px;
```

```
          outline: none;
        }
      }

      .price::before {
        content: '¥';
        font-size: 22px;
      }
    }
</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

```
// 商品列表
  const goodsList = [
    {
      id: '4001172',
      name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
      price: 289,
      picture:
        'https://yanxuan-item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg'
    },
    {
      id: '4001594',
      name: '日式黑陶功夫茶组双侧把茶具礼盒装',
      price: 288,
      picture:
        'https://yanxuan-item.nosdn.127.net/3346b7b92f9563c7a7e24c7ead883f18.jpg'
    },
    {
      id: '4001009',
      name: '竹制干泡茶盘正方形沥水茶台品茶盘',
      price: 109,
      picture:
        'https://yanxuan-
item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png'
    },
    {
      id: '4001874',
      name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',
      price: 488,
      picture:
        'https://yanxuan-
item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png'
    },
    {
      id: '4001649',
      name: '大师监制龙泉青瓷茶叶罐',
      price: 139,
      picture:
        'https://yanxuan-
item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png'
    },
    {
      id: '3997185',
      name: '与众不同的口感汝瓷白酒杯套组1壶4杯',
      price: 108,
      picture:
        'https://yanxuan-
item.nosdn.127.net/8e21c794dfd3a4e8573273ddae50bce2.jpg'
    },
    {
      id: '3997403',
      name: '手工吹制更厚实白酒杯壶套装6壶6杯',
      price: 100,
      picture:
        'https://yanxuan-
item.nosdn.127.net/af2371a65f60bce152a61fc22745ff3f.jpg'
    },
    {
      id: '3998274',
      name: '德国百年工艺高端水晶玻璃红酒杯2支装',
      price: 139,
      picture:
```

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

3.3 代码示例

App.vue

```
<script setup>
  import MyGoods from './components/MyGoods.vue'
  // 商品列表
  const goodsList = [
    {
      id: '4001172',
      name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
      price: 289,
      picture:
        'https://yanxuan-item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg'
    },
    {
      id: '4001594',
      name: '日式黑陶功夫茶组双侧把茶具礼盒装',
      price: 288,
      picture:
        'https://yanxuan-item.nosdn.127.net/3346b7b92f9563c7a7e24c7ead883f18.jpg'
    },
    {
      id: '4001009',
      name: '竹制干泡茶盘正方形沥水茶台品茶盘',
      price: 109,
      picture:
        'https://yanxuan-item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png'
    },
    {
      id: '4001874',
      name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',
      price: 488,
      picture:
        'https://yanxuan-item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png'
    },
    {
      id: '4001649',
      name: '大师监制龙泉青瓷茶叶罐',
      price: 139,
      picture:
        'https://yanxuan-
item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png'
    },
    {
      id: '3997185',
      name: '与众不同的口感汝瓷白酒杯套组1壶4杯',
      price: 108,
      picture:
        'https://yanxuan-
item.nosdn.127.net/8e21c794dfd3a4e8573273ddae50bce2.jpg'
    },
    {
      id: '3997403',
      name: '手工吹制更厚实白酒杯壶套装6壶6杯',
      price: 100,
      picture:
        'https://yanxuan-
item.nosdn.127.net/af2371a65f60bce152a61fc22745ff3f.jpg'
    },
    {
      id: '3998274',
      name: '德国百年工艺高端水晶玻璃红酒杯2支装',
      price: 139,
      picture:
        'https://yanxuan-
item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'
    }
  ]
</script>
<template>
  <div class="list">
     <!-- 循环遍历、独立传值-->
     <MyGoods
         v-for="item in goodsList"
         :key="item.id"
         :imgUrl="item.picture"
         :title="item.name"
         :price="item.price"
     />
  </div>
</template>

<style lang="scss">
```

```
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  .list {
    width: 1000px;
    margin: 0 auto;
    display: flex;
    flex-wrap: wrap;
  }
</style>
```

MyGoods.vue

```
<script setup>
  defineProps(['imgUrl', 'title', 'price'])
</script>

<template>
  <div class="item">
    <img
      :src="imgUrl"
      :alt="title" />
    <p class="name">{{ title}}</p>
    <p class="price">
      <span>{{ price }}</span>
      <button>砍一刀</button>
    </p>
  </div>
</template>

<style lang="scss" scoped>
  .item {
    width: 240px;
    margin-left: 10px;
    padding: 20px 30px;
    transition: all 0.5s;
    margin-bottom: 20px;
    .item:nth-child(4n) {
      margin-left: 0;
    }

    &:hover {
      box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
      transform: translate3d(0, -4px, 0);
      cursor: pointer;
    }

    img {
      width: 100%;
    }

    .name {
      font-size: 18px;
      margin-bottom: 10px;
      color: #666;
    }

    .price {
      display: flex;
      height: 36px;
      align-items: center;
      font-size: 22px;
      color: firebrick;
      button {
        margin-left: 80px;
        font-size: 14px;
        outline: none;
      }
    }

    .price::before {
      content: '¥';
      font-size: 22px;
    }
  }
</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?

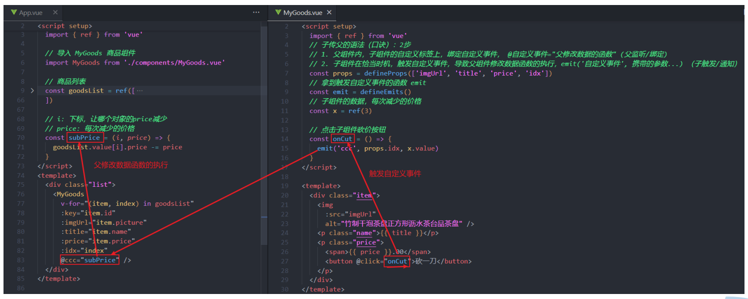
4.2 解决方案

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）



```
<script setup>
  import { ref } from 'vue'

  // 导入 MyGoods 商品组件
  import MyGoods from './components/MyGoods.vue'

  // 商品列表
  const goodsList = ref([…
  ])

  // i: 下标，让哪个对象的price减少
  // price: 每次减少的价格
  const subPrice = (i, price) => {
    goodsList.value[i].price -= price
  }
</script>
<template>
  <div class="list">
    <MyGoods
      v-for="(item, index) in goodsList"
      :key="item.id"
      :imgUrl="item.picture"
      :title="item.name"
      :price="item.price"
      :idx="index"
      @ccc="subPrice" />
  </div>
</template>
```

```
<script setup>
  import { ref } from 'vue'
  // 子传父的语法（口诀）：2步
  // 1. 父组件内，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）
  // 2. 子组件在恰当时机，触发自定义事件，导致父组件修改数据函数的执行，emit('自定义事件', 携带的参数...) （子触发/通知）
  const props = defineProps(['imgUrl', 'title', 'price', 'idx'])
  // 拿到触发自定义事件的函数 emit
  const emit = defineEmits()
  // 子组件的数据，每次减少的价格
  const x = ref(3)

  // 点击子组件砍价按钮
  const onCut = () => {
    emit('ccc', props.idx, x.value)
  }
</script>

<template>
  <div class="item">
    <img
      :src="imgUrl"
      alt="竹制干泡茶盘正方形沥水茶台品茶盘" />
    <p class="name">{{ title }}</p>
    <p class="price">
      <span>{{ price }}.00</span>
      <button @click="onCut">砍一刀</button>
    </p>
  </div>
</template>
```

4.4 代码示例

App.vue

```
<script setup>
  import { ref } from 'vue'
  // 导入 MyGoods 商品组件
  import MyGoods from './components/MyGoods.vue'
  // 商品列表
  const goodsList = ref([
    {
      id: '4001172',
      name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
      price: 289,
      picture:
        'https://yanxuan-
item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg'
    },
    {
      id: '4001594',
      name: '日式黑陶功夫茶组双侧把茶具礼盒装',
      price: 288,
      picture:
        'https://yanxuan-item.nosdn.127.net/3346b7b92f9563c7a7e24c7ead883f18.jpg'
    },
    {
      id: '4001009',
      name: '竹制干泡茶盘正方形沥水茶台品茶盘',
      price: 109,
      picture:
        'https://yanxuan-item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png'
    },
    {
      id: '4001874',
      name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',
      price: 488,
      picture:
        'https://yanxuan-item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png'
    },
    {
      id: '4001649',
      name: '大师监制龙泉青瓷茶叶罐',
      price: 139,
      picture:
        'https://yanxuan-item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png'
    },
    {
      id: '3997185',
      name: '与众不同的口感汝瓷白酒杯套组1壶4杯',
      price: 108,
      picture:
        'https://yanxuan-item.nosdn.127.net/8e21c794dfd3a4e8573273ddae50bce2.jpg'
    },
    {
      id: '3997403',
      name: '手工吹制更厚实白酒杯壶套装6壶6杯',
      price: 100,
      picture:
        'https://yanxuan-item.nosdn.127.net/af2371a65f60bce152a61fc22745ff3f.jpg'
    },
    {
      id: '3998274',
      name: '德国百年工艺高端水晶玻璃红酒杯2支装',
      price: 139,
      picture:
        'https://yanxuan-item.nosdn.127.net/8896b897b3ec6639bbd1134d66b9715c.jpg'
    }
  ])

  // i: 下标，让哪个对象的 price 减少
  // price: 每次减少的价格
  const subPrice = (i, price) => {
    goodsList.value[i].price -= price
  }
</script>
<template>
  <div class="list">
    <MyGoods
      v-for="(item, index) in goodsList"
      :key="item.id"
      :imgUrl="item.picture"
      :title="item.name"
      :price="item.price"
      :idx="index"
      @ccc="subPrice" />
  </div>
</template>

<style lang="scss">
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  .list {
    width: 1000px;
    margin: 0 auto;
    display: flex;
    flex-wrap: wrap;
  }
</style>
```

MyGoods.vue

```
<script setup>
  import { ref } from 'vue'

  // 子传父的语法（口诀）：2步
  // 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)
  // 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）

  const props = defineProps(['imgUrl', 'title', 'price', 'idx'])

  // 拿到触发自定义事件的函数 emit
  const emit = defineEmits()

  // 子组件的数据，表示每次减少的价格
  const x = ref(3)

  // 点击子组件砍价按钮
  const onCut = () => {
    // 触发 ccc 自定义事件
    emit('ccc', props.idx, x.value)
  }
</script>

<template>
  <div class="item">
    <img
      :src="imgUrl"
      alt="竹制干泡茶盘正方形沥水茶台品茶盘" />
    <p class="name">{{ title }}</p>
    <p class="price">
      <span>{{ price }}.00</span>
      <button @click="onCut">砍一刀</button>
    </p>
  </div>
</template>

<style scoped lang="scss">
  .item {
    width: 240px;
    margin-left: 10px;
    padding: 20px 30px;
    transition: all 0.5s;
    margin-bottom: 20px;
    .item:nth-child(4n) {
      margin-left: 0;
    }

    &:hover {
      box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
      transform: translate3d(0, -4px, 0);
      cursor: pointer;
    }

    img {
      width: 100%;
    }

    .name {
      font-size: 18px;
      margin-bottom: 10px;
      color: #666;
    }

    .price {
      display: flex;
      align-items: center;
      height: 36px;
      font-size: 22px;
      color: firebrick;
      button {
        margin-left: 50px;
        font-size: 14px;
      }
    }

    .price::before {
      content: '¥';
      font-size: 22px;
    }
  }
</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：

```
<MyGoods
  :imgUrl="goodsObj.picture"
  :title="goodsObj.name"
  :price="goodsObj.price" />
```

1.2 思考

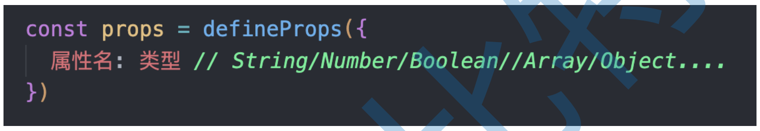
组件的props可以乱传吗?

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

```
<script setup></script>
<template>
  <div class="my-progress">
    <div
      class="inner"
      style="width:50%">
      <span>50%</span>
    </div>
  </div>
</template>

<style scoped>
```

```
  .my-progress {
    height: 26px;
    width: 400px;
    border-radius: 15px;
    background-color: #272425;
    border: 3px solid #272425;
    box-sizing: border-box;
    margin-bottom: 30px;
  }
  .inner {
    position: relative;
    background: #379bff;
    border-radius: 15px;
    height: 25px;
    box-sizing: border-box;
    left: -3px;
    top: -2px;
  }
  .inner span {
    position: absolute;
    right: -30px;
    top: 26px;
  }
</style>
```

1.6 代码演示

App.vue

```
<script setup>
  import { ref } from 'vue'
  import MyProgress from './components/MyProgress.vue'
  const width = ref(50)
</script>
<template>
  <div class="app">
    <MyProgress :width="width" />
  </div>
</template>

<style></style>
```

MyProgress.vue

```
<script setup>
  defineProps(['width'])
</script>
<template>
  <div class="my-progress">
    <div
      class="inner"
      :style="{ width: width + '%' }">
      <span>{{ width }}%</span>
    </div>
  </div>
</template>

<style scoped>
  .my-progress {
    height: 26px;
    width: 400px;
    border-radius: 15px;
    background-color: #272425;
    border: 3px solid #272425;
    box-sizing: border-box;
    margin-bottom: 30px;
  }
  .inner {
    position: relative;
    background: #379bff;
    border-radius: 15px;
    height: 25px;
    box-sizing: border-box;
    left: -3px;
    top: -2px;
  }
  .inner span {
    position: absolute;
    right: -30px;
    top: 26px;
  }
</style>
```

2. 完整写法

2.1 语法

```
defineProps({
  属性名: {
    type: 类型,  // Number String Boolean ...
    required: true, // 是否必填
    default: 默认值, // 默认值

    // value: 父传子的值
    validator (value) {
      // 自定义校验逻辑
      return 布尔值
    }
  }
})
```

2.2 代码示例

```
<script setup>
  defineProps({
    width: {
      type: Number,
      // required: true
      default: 0,
      validator: (value) => {
        if (value >= 0 && value <= 100) {
          return true
        } else {
          console.error('The width prop must be between 0 and 100')
          return false
        }
      }
    }
  })
</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

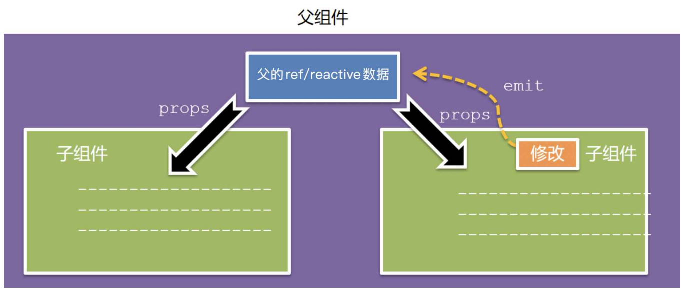
3.2 区别

- 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)

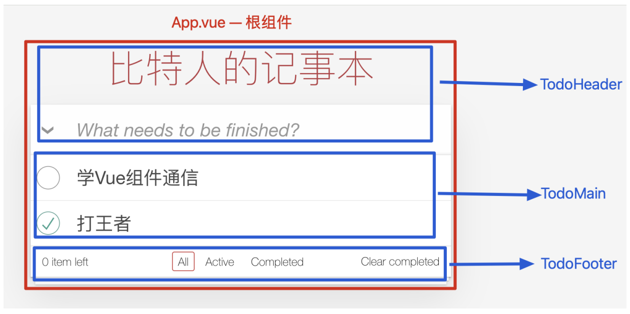
   2、不同点：ref/reactive数据是组件自己的(可读可写)，props数据是父给的(只读的)

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`

```
<script setup></script>

<template>
  <header class="header">
    <h1>比特人的记事本</h1>
    <input class="new-todo" placeholder="What needs to be finished?" autofocus />
  </header>
</template>
```

`components/TodoMain.vue`

```
<script setup></script>

<template>
  <section class="main">
    <input id="toggle-all" class="toggle-all" type="checkbox" />
    <label for="toggle-all">Mark all as complete</label>
    <ul class="todo-list">
      <li>
        <div class="view">
          <input class="toggle" type="checkbox" />
          <label>学Vue组件通信</label>
          <button class="destroy"></button>
        </div>
      </li>
      <li>
        <div class="view">
          <input class="toggle" type="checkbox" checked />
          <label>打王者</label>
          <button class="destroy"></button>
        </div>
      </li>
    </ul>
  </section>
</template>
```

`components/TodoFooter.vue`

```html
<script setup></script>

<template>
  <footer class="footer">
    <span class="todo-count"><strong>0</strong> item left</span>
    <ul class="filters">
      <li>
        <a href="#/" class="selected">All</a>
      </li>
      <li>
        <a href="#/active">Active</a>
      </li>
      <li>
        <a href="#/completed">Completed</a>
      </li>
    </ul>
    <button class="clear-completed">
      Clear completed
    </button>
  </footer>
</template>
```

`assets/style.css`

```css
@charset 'utf-8';

html,
body {
  margin: 0;
  padding: 0;
}

button {
  margin: 0;
  padding: 0;
  border: 0;
  background: none;
  font-size: 100%;
  vertical-align: baseline;
  font-family: inherit;
  font-weight: inherit;
  color: inherit;
  -webkit-appearance: none;
  appearance: none;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

body {
  font:
    14px 'Helvetica Neue',
    Helvetica,
    Arial,
    sans-serif;
  line-height: 1.4em;
  background: #f5f5f5;
  color: #111111;
  min-width: 230px;
  max-width: 550px;
  margin: 0 auto;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  font-weight: 300;
}

.hidden {
  display: none;
}

.todoapp {
  background: #fff;
  margin: 130px 0 40px 0;
  position: relative;
  box-shadow:
    0 2px 4px 0 rgba(0, 0, 0, 0.2),
    0 25px 50px 0 rgba(0, 0, 0, 0.1);
}

.todoapp input::-webkit-input-placeholder {
  font-style: italic;
  font-weight: 400;
  color: rgba(0, 0, 0, 0.4);
}

.todoapp input::-moz-placeholder {
  font-style: italic;
  font-weight: 400;
  color: rgba(0, 0, 0, 0.4);
}

.todoapp input::input-placeholder {
  font-style: italic;
  font-weight: 400;
  color: rgba(0, 0, 0, 0.4);
}

.todoapp h1 {
  position: absolute;
  top: -100px;
  width: 100%;
  font-size: 50px;
  font-weight: 200;
  text-align: center;
  color: #b83f45;
  -webkit-text-rendering: optimizeLegibility;
  -moz-text-rendering: optimizeLegibility;
  text-rendering: optimizeLegibility;
}

.new-todo,
.edit {
  position: relative;
  margin: 0;
  width: 100%;
  font-size: 24px;
  font-family: inherit;
  font-weight: inherit;
  line-height: 1.4em;
  color: inherit;
  padding: 6px;
  border: 1px solid #999;
  box-shadow: inset 0 -1px 5px 0 rgba(0, 0, 0, 0.2);
  box-sizing: border-box;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.new-todo {
  padding: 16px 16px 16px 60px;
  height: 65px;
  border: none;
  background: rgba(0, 0, 0, 0.003);
  box-shadow: inset 0 -2px 1px rgba(0, 0, 0, 0.03);
}

.main {
  position: relative;
  z-index: 2;
  border-top: 1px solid #e6e6e6;
```

```css
}

.toggle-all {
  width: 1px;
  height: 1px;
  border: none; /* Mobile Safari */
  opacity: 0;
  position: absolute;
  right: 100%;
  bottom: 100%;
}

.toggle-all + label {
  display: flex;
  align-items: center;
  justify-content: center;
  width: 45px;
  height: 65px;
  font-size: 0;
  position: absolute;
  top: -65px;
  left: -0;
}

.toggle-all + label:before {
  content: ' ';
  display: inline-block;
  font-size: 22px;
  color: #949494;
  padding: 10px 27px 10px 27px;
  -webkit-transform: rotate(90deg);
  transform: rotate(90deg);
}

.toggle-all:checked + label:before {
  color: #484848;
}

.todo-list {
  margin: 0;
  padding: 0;
  list-style: none;
}

.todo-list li {
  position: relative;
  font-size: 24px;
  border-bottom: 1px solid #ededed;
}

.todo-list li:last-child {
  border-bottom: none;
}

.todo-list li.editing {
  border-bottom: none;
  padding: 0;
}

.todo-list li.editing .edit {
  display: block;
  width: calc(100% - 43px);
  padding: 12px 16px;
  margin: 0 0 0 43px;
}

.todo-list li.editing .view {
  display: none;
}

.todo-list li .toggle {
  text-align: center;
  width: 40px;
  /* auto, since non-WebKit browsers doesn't support input styling */
  height: auto;
  position: absolute;
  top: 0;
  bottom: 0;
  margin: auto 0;
  border: none; /* Mobile Safari */
  -webkit-appearance: none;
  appearance: none;
}

.todo-list li .toggle {
  opacity: 0;
}

.todo-list li .toggle + label {
  /*
    Firefox requires `#` to be escaped - 
https://bugzilla.mozilla.org/show_bug.cgi?id=922433
    IE and Edge requires *everything* to be escaped to render, so we do that 
instead of just the `#` - https://developer.microsoft.com/en-us/microsoft-
```

```
edge/platform/issues/7157459/
   */
  background-image: url('data:image/svg+xml;utf8,%3Csvg%20xmlns%3D%22http%3A//www.w3.org/2000/svg%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%23949494%22%20stroke-width%3D%223%22/%3E%3C/svg%3E');
  background-repeat: no-repeat;
  background-position: center left;
}

.todo-list li .toggle:checked + label {
  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');
}

.todo-list li label {
  overflow-wrap: break-word;
  padding: 15px 15px 15px 60px;
  display: block;
  line-height: 1.2;
  transition: color 0.4s;
  font-weight: 400;
  color: #484848;
}

.todo-list li.completed label {
  color: #949494;
  text-decoration: line-through;
}

.todo-list li .destroy {
  display: none;
  position: absolute;
  top: 0;
  right: 10px;
  bottom: 0;
  width: 40px;
  height: 40px;
  margin: auto 0;
  font-size: 30px;
```

```css
  color: #949494;
  transition: color 0.2s ease-out;
}

.todo-list li .destroy:hover,
.todo-list li .destroy:focus {
  color: #c18585;
}

.todo-list li .destroy:after {
  content: '×';
  display: block;
  height: 100%;
  line-height: 1.1;
}

.todo-list li:hover .destroy {
  display: block;
}

.todo-list li .edit {
  display: none;
}

.todo-list li.editing:last-child {
  margin-bottom: -1px;
}

.footer {
  padding: 10px 15px;
  height: 20px;
  text-align: center;
  font-size: 15px;
  border-top: 1px solid #e6e6e6;
}

.footer:before {
  content: '';
  position: absolute;
  right: 0;
  bottom: 0;
  left: 0;
  height: 50px;
  overflow: hidden;
  box-shadow:
    0 1px 1px rgba(0, 0, 0, 0.2),
    0 8px 0 -3px #f6f6f6,
    0 9px 1px -3px rgba(0, 0, 0, 0.2),
    0 16px 0 -6px #f6f6f6,
    0 17px 2px -6px rgba(0, 0, 0, 0.2);
}

.todo-count {
  float: left;
  text-align: left;
}

.todo-count strong {
  font-weight: 300;
}

.filters {
  margin: 0;
  padding: 0;
  list-style: none;
  position: absolute;
  right: 0;
  left: 0;
}

.filters li {
  display: inline;
}

.filters li a {
  color: inherit;
  margin: 3px;
  padding: 3px 7px;
  text-decoration: none;
  border: 1px solid transparent;
  border-radius: 3px;
}

.filters li a:hover {
  border-color: #db7676;
}

.filters li a.selected {
  border-color: #ce4646;
}

.clear-completed,
html .clear-completed:active {
  float: right;
  position: relative;
  line-height: 19px;
  text-decoration: none;
  cursor: pointer;
}

.clear-completed:hover {
  text-decoration: underline;
}

.info {
  margin: 65px auto 0;
  color: #4d4d4d;
  font-size: 11px;
  text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);
  text-align: center;
}

.info p {
  line-height: 1;
}

.info a {
  color: inherit;
  text-decoration: none;
  font-weight: 400;
}

.info a:hover {
  text-decoration: underline;
}

/*
  Hack to remove background from Mobile Safari.
  Can't use it globally since it destroys checkboxes in Firefox
*/
@media screen and (-webkit-min-device-pixel-ratio: 0) {
  .toggle-all,
  .todo-list li .toggle {
    background: none;
  }

  .todo-list li .toggle {
    height: 40px;
  }
}

```

```
@media (max-width: 430px) {
  .footer {
    height: 50px;
  }

  .filters {
    bottom: 10px;
  }
}

:focus,
.toggle:focus + label,
.toggle-all:focus + label {
  box-shadow: 0 0 2px 2px #cf7d7d;
  outline: 0;
}
```

`App.vue`

```
<script setup>
import { ref } from 'vue'
import './asssts/style.css'
import TodoHeader from './components/TodoHeader.vue'
import TodoMain from './components/TodoMain.vue'
import TodoFooter from './components/TodoFooter.vue'

const todoList = ref([
  { id: 321, name: '学Vue组件通信', finished: false },
  { id: 127, name: '打王者', finished: true },
  { id: 666, name: '跑步1小时', finished: false }
])
</script>

<template>
  <section class="todoapp">
    <todo-header />
    <todo-main />
    <todo-footer />
  </section>
</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`

```
<script setup>
const props = defineProps({
  // 数据列表
  todoList: {
    type: Array,
    default: () => []
  }
})
</script>

<template>
  <section class="main">
    <input id="toggle-all" class="toggle-all" type="checkbox" />
    <label for="toggle-all">Mark all as complete</label>
    <ul class="todo-list">
      <li v-for="item in props.todoList"
          :key="item.id"
          :class="{ completed: item.finished }">
        <div class="view">
          <input class="toggle"
                 type="checkbox"
                 v-model="item.finished" />
          <label>{{ item.name }}</label>
          <button class="destroy"></button>
        </div>
      </li>
    </ul>
  </section>
</template>
```

`App.vue`

```
<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`

```
<script setup>
import { ref } from 'vue'

const emit = defineEmits()

// 任务名称
const title = ref('')

// 回车
const onEnter = () => {
  // 非空校验
  if (!title.value) return alert('任务名称不能为空')
  // 通知父组件
  emit('add-todo', title.value)
  // 清空输入框
  title.value = ''
}
</script>

<template>
  <header class="header">
    <h1>比特人的记事本</h1>
    <input
      class="new-todo"
      placeholder="What needs to be finished?"
      autofocus
      v-model.trim="title"
      @keyup.enter="onEnter"
    />
  </header>
</template>

```

`App.vue`

```
<script setup>
// ...

// 添加
const addTodo = (name) => {
  // 添加至数组头部
  todoList.value.unshift({
    id: Date.now(), // 时间戳
    name,
    finished: false
  })
}
</script>

<template>
  <section class="todoapp">
    <todo-header @add-todo="addTodo" />
  </section>
</template>

```

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`

```
<script setup>
  const emit = defineEmits()
</script>

<template>
  <section class="main">
    <ul class="todo-list">
```

```
      <li
        v-for="(item, index) in props.todoList">
        <div class="view">
          <button @click="emit('del-todo', index)"></button>
        </div>
      </li>
    </ul>
  </section>
</template>
```

`App.vue`

```
<script setup>

// 删除
const delTodo = (i) => {
  if (window.confirm('确认删除么?')) {
    todoList.value.splice(i, 1)
  }
}
</script>

<template>
  <todo-main @del-todo="delTodo" />
</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`

```
<script setup>
const props = defineProps({
  todoList: {
    type: Array,
    default: () => []
  }
})
const emit = defineEmits()
</script>

<template>
  <footer class="footer">
    <span class="todo-count"
      ><strong>{{ props.todoList.length }}</strong> item left</span
    >
    <button class="clear-completed" @click="emit('clear-todo')">Clear completed</button>
  </footer>
</template>
```

`App.vue`

```
<script setup>
import { watch } from 'vue'
// 本地存储键名
const TODO_LIST_KEY = 'todo-list-key'
const todoList = ref(
  // 取; 有则用、否则用默认值
  JSON.parse(localStorage.getItem(TODO_LIST_KEY)) || [
    { id: 321, name: '学Vue组件通信', finished: false },
    { id: 127, name: '打王者', finished: true },
    { id: 666, name: '跑步1小时', finished: false }
  ]
)

// 清除已完成
const clearTodo = () => {
  todoList.value = todoList.value.filter((todo) => !todo.finished)
}

// 监视 todoList 的变化
watch(
  todoList,
  (newVal) => {
    // 存
    localStorage.setItem(TODO_LIST_KEY, JSON.stringify(newVal))
  },
  {// 深度监视
    deep: true
  }
)
</script>

<template>
  <todo-footer :todoList="todoList" @clear-todo="clearTodo" />
</template>
```

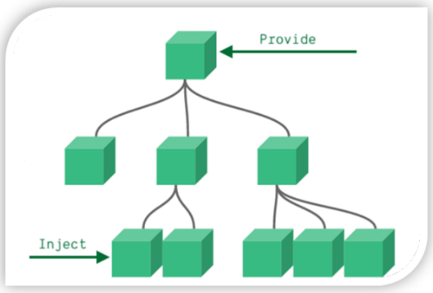
五、非父子组件通信

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

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

2、后代(子/孙/曾孙...)组件通过 **inject** 获取数据

```
import { inject } from 'vue'
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)

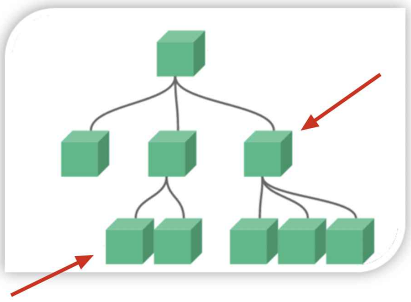
   2、后代组件: const value = inject('数据名称')

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)

```
yarn add mitt -S
```

```
import mitt from 'mitt'
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

```
<script setup>

</script>

<template>
  <div
    class="my-left"
    style="
      flex: 1;
      align-items: center;
      height: 100px;
      background: plum;
      text-align: center;
    ">
    <h3>我是牛郎，点击发送消息给织女</h3>
    <button>Send Message</button>
  </div>
</template>

<style></style>
```

1. 确定消息接收方组件
2. `meipo.on('消息名称', (...args) => { })`
3. MyRight.vue

```
<script setup>

</script>

<template>
  <div
    class="my-right"
    style="
      flex: 1;
      align-items: center;
      height: 100px;
      background: paleturquoise;
      text-align: center;
    ">
    <h3>我是织女，收到来自牛郎的消息：xxx </h3>
  </div>
</template>

<style></style>
```

1. App.vue

```
<script setup>
  import MyLeft from './components/MyLeft.vue'
  import MyRight from './components/MyRight.vue'
</script>
<template>
  <div class="container">
    <MyLeft />
    <MyRight />
  </div>
</template>

<style>
  .container {
    display: flex;
  }
</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()`