Vue组件基础

创建组件_组件通信_todo案例

目录

Contents

学习目标

Learning Objectives

1. 能够理解Vue组件概念和作用
2. 能够掌握封装创建组件能力
3. 能够使用组件之间通信
4. 能够完成todo案例

目录

Contents

Vue组件创建和使用

问题1：以前遇到重复的标签，如何做的？

问题2：效率或独立性高吗，担心变量重名？

1.0_折叠面板-实现多个


```
<template>
  <div id="app">
    <h3>案例：折叠面板</h3>
    <div>
      <div class="title">
        <h4>芙蓉楼送辛渐</h4>
        <span class="btn" @click="isShow = !isShow">
          {{ isShow ? '收起' : '展开' }}
        </span>
      </div>
      <div class="container" v-show="isShow">
        <p>寒雨连江夜入吴，</p>
        <p>平明送客楚山孤。</p>
        <p>洛阳亲友如相问，</p>
        <p>一片冰心在玉壶。</p>
      </div>
    </div>
  </div>
</template>
```

1.0_折叠面板-实现多个

- 多套相同结构标签, 变量区分开, 分别控制, 这样好吗?

```
data() {
  return {
    isShow: false,
    isShow1: false,
    isShow2: false
  }
}
```

- 解决方案: 采用vue提供的单.vue文件-组件方式来封装一套然后复用

```
<template>
  <div id="app">
    <h3>案例：折叠面板</h3>
    <Pannel></Pannel>
    <Pannel></Pannel>
    <Pannel></Pannel>
  </div>
</template>
```

```
<template>
  <div>
    <div class="title">
      <h4>芙蓉楼送辛渐</h4>
      <span class="btn" @click="isShow = !isShow">
        {{ isShow ? "收起" : "展开" }}
      </span>
    </div>
    <div class="container" v-show="isShow">
      <p>寒雨连江夜入吴,</p>
      <p>平明送客楚山孤。</p>
      <p>洛阳亲友如相问，</p>
      <p>一片冰心在玉壶。</p>
    </div>
  </div>
</template>

<script>
export default {
  name: "Pannel",
  data() {
    return {
      isShow: false,
    }
  }
}
</script>
```

1. 遇到重复标签想复用?

   封装成组件

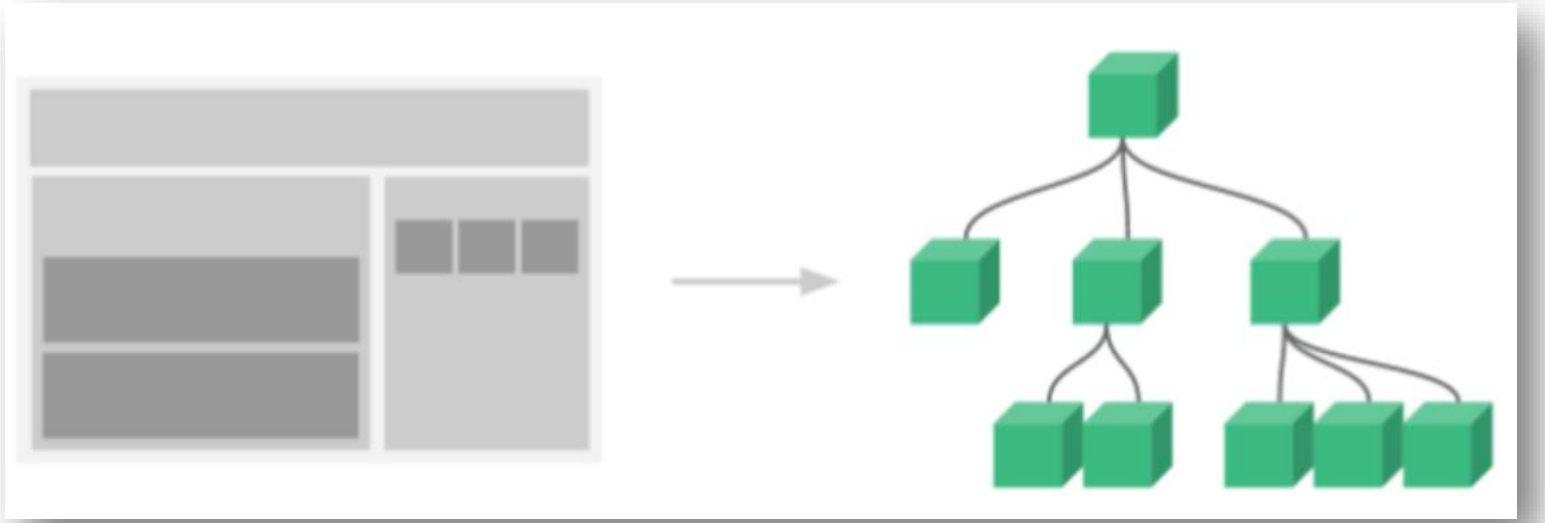
2. 组件好处?

   各自独立, 便于复用

问题：组件的概念？

1.1_组件概念

- 组件是可复用的 Vue 实例, 封装标签, 样式和JS代码
- 组件化： 封装的思想，把页面上 `可重用的部分` 封装为 `组件`，从而方便项目的 开发 和 维护
- 一个页面， 可以拆分成一个个组件，一个组件就是一个整体, 每个组件可以有自己独立的 结构 样式 和 行为(html, css和js)



组件是什么?

可复用的vue实例, 封装标签, 样式, JS

什么时候封装组件?

遇到重复标签, 可复用的时候

组件好处?

各自独立, 互不影响

问题：如何创建和使用组件？

1.2_组件_基础使用

目标：每个组件都是一个独立的个体，代码里体现为一个独立的.vue文件

1. 创建组件，封装要复用的标签，样式，JS代码
2. 注册组件
   - 全局注册 – main.js中　　– 语法如图
   - 局部注册 – 某.vue文件内 – 语法如图
3. 使用组件

```
import Vue from 'vue'
import 组件对象 from 'vue文件路径'

Vue.component("组件名", 组件对象)
```

```
<template>
  <div id="app">
    <h3>案例：折叠面板</h3>
    <组件名></组件名>
    <组件名></组件名>
  </div>
</template>
```

```
import 组件对象 from 'vue文件路径'

export default {
    components: {
        "组件名": 组件对象
    }
}
```

创建和使用组件步骤?

创建.vue文件 – 标签 – 样式 – JS进去

注册组件 (全局 / 局部)

使用组件 (组件名用作标签)

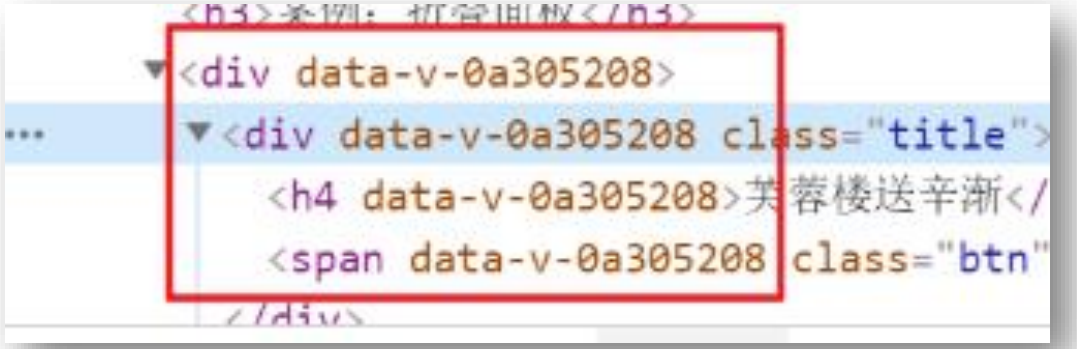
组件运行结果?

把组件标签最终替换成, 封装的组件内标签

问题：组件内的scoped是如何工作的?

1.3_组件-scoped作用

- 准备: 当前组件内标签都被添加 data-v-hash值 的属性
- 获取: css选择器都被添加 [data-v-hash值] 的属性选择器



Vue组件内样式, 只针对当前组件内标签生效如何做?

给style上添加scoped

原理和过程是什么?

会自动给标签添加data-v-hash值属性, 所有选择都带属性选择

目录

Contents

Vue组件通信

问题1: 从一个组件内, 给另外一个组件传值?

问题2: 不使用localStorage, 直接传递, 如何做?

2.0_组件通信_父传子_props

目标：父组件 -> 子组件 传值

- 首先明确父和子是谁, 在父引入子 (被引入的是子)
  - 父: App.vue
  - 子: MyProduct.vue
- 创建MyProduct.vue如下图所示

2.0_组件通信_父传子_props

目标：父组件 -> 子组件 传值

1. 子组件内, 定义变量, 准备接收, 然后使用变量

```
<template>
  <div class="my-product">
    <h3>标题: {{ title }}</h3>
    <p>价格: {{ price }}元</p>
    <p>{{ info }}</p>
  </div>
</template>

<script>
export default {
  props: ['title', 'price', 'info']
}
</script>
```

2.0_组件通信_父传子_props

目标：父组件 -> 子组件 传值

2. 父组件(App.vue)内, 要展示封装的子组件(MyProduct.vue)

引入组件, 注册组件, 使用组件, 传值进去

```vue
<template>
  <div>
    <MyProduct title="超级好吃的口水鸡" price="50" :info="msg"></MyProduct>
  </div>
</template>
<script>
import MyProduct from './components/MyProduct'
export default {
  data(){
    return {
      msg: '开业大酬宾, 全场8折'
    }
  },
  components: {
    MyProduct
  }
}
</script>
```

什么时候需要父传子技术?

从一个vue组件里把值传给另一个vue组件(父->子)

父传子口诀(步骤)是什么?

子组件内, props定义变量, 在子组件使用变量

父组件内, 使用子组件, 属性方式给props变量传值

问题1: 能循环使用组件吗?

问题2: 每次循环使用组件, 独立向组件内传值?

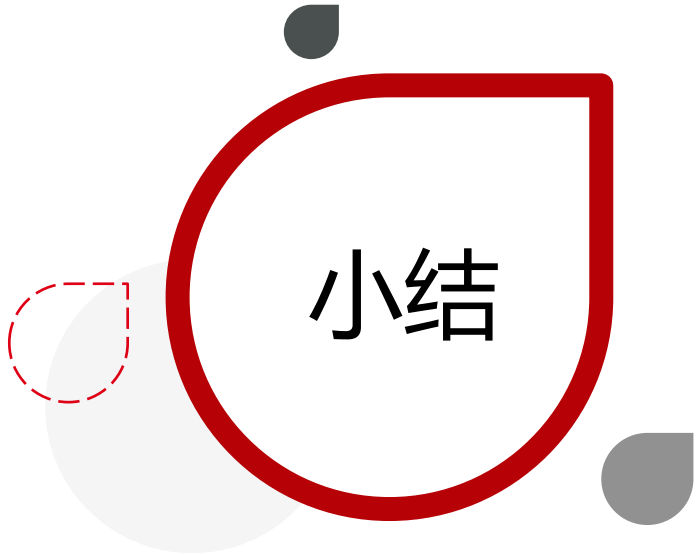
2.1_组件通信_父向子-配合循环

目标：父组件 -> 子组件 循环使用-传值

```
<template>
  <div>
    <MyProduct v-for="obj in list" :key="obj.id"
    :title="obj.proname"
    :price="obj.proprice"
    :info="obj.info"></MyProduct>
  </div>
</template>
```

每次循环obj和组件都是独立的，新的

```
data() {
  return {
    list: [
      { id: 1, proname: "超级好吃的棒棒糖", proprice: 18.8, info: '开业大酬宾，全场8折' },
      { id: 2, proname: "超级好吃的大鸡腿", proprice: 34.2, info: '好吃不腻，快来买啊' },
      { id: 3, proname: "超级无敌的冰激凌", proprice: 14.2, info: '炎热的夏天，来个冰激凌了' }
    ]
  }
}
```



循环使用组件注意事项?

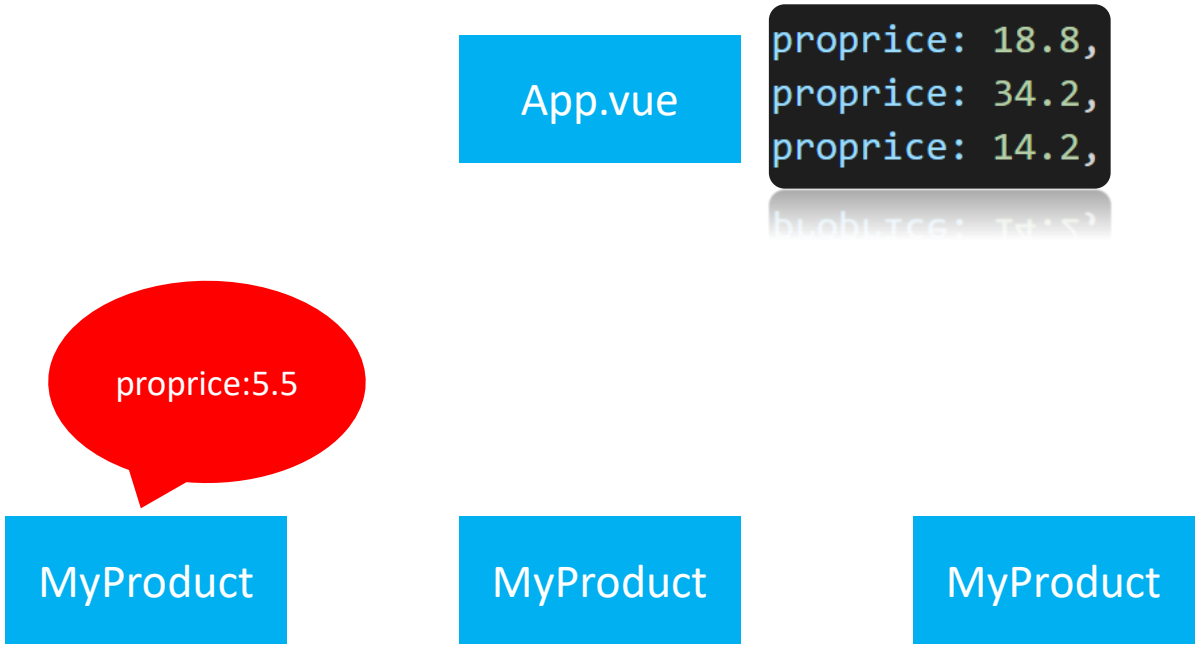
每次循环, 变量和组件, 都是独立的

问题1: 子组件内想实现砍价功能, 点一次按钮砍掉价格?

问题2: 子组件内能改变, 父传入的数据吗?

2.2_单向数据流

目标：从父到子的数据流向，叫单向数据流



原因：子组件修改，不通知父级，造成数据不一致性

Vue规定props里的变量，本身是只读的

为何不建议，子组件修改父组件传过来的值？

父子数据不一致，而且子组件是依赖父传入的值

什么是单向数据流？

从父到子的数据流向，叫单向数据流

props里定义的变量能修改吗？

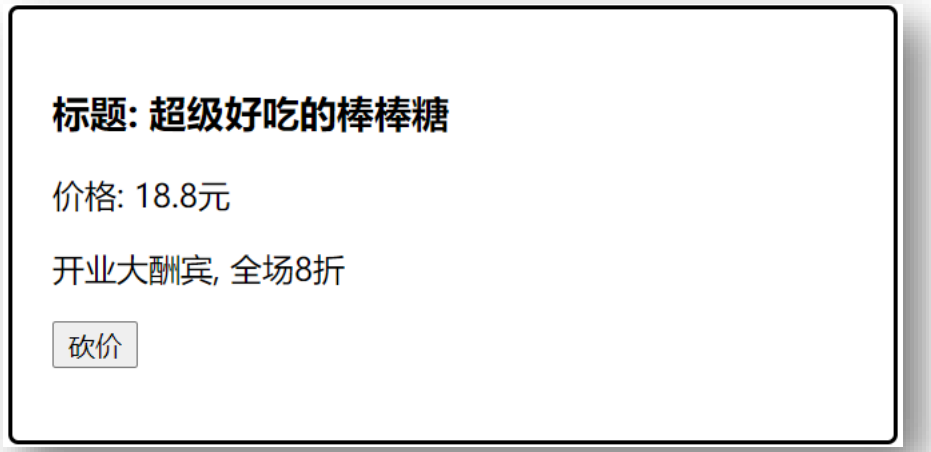
不能，props里的变量本身是只读的

问题: 那子组件如何才能修改父组件里的数据呢?

2.3_组件通信_子向父_自定义事件

目标：子组件触发父自定义事件方法

- 需求: 商品组件, 实现砍价功能



- 前置补充, 父 -> 索引 -> 子组件 (用于区分哪个子组件)

```
<MyProduct
  v-for="(obj, index) in list" :key="obj.id"
  :title="obj.proname"
  :price="obj.proprice"
  :info="obj.info"
  :index="index"
></MyProduct>
```

2.3_组件通信_子向父_自定义事件

目标：子组件触发父自定义事件方法

1. 父组件内, 绑定自定义事件和事件处理函数

   语法: @自定义事件名="父methods里函数名"

```
<div>
  <MyProduct
    v-for="(obj, index) in list" :key="obj.id"
    :title="obj.proname"
    :price="obj.proprice"
    :info="obj.info"
    :index="index"
    @subprice="fn"
  ></MyProduct>
</div>
```

```
methods: {
  fn(index, price) {
    this.list[index].proprice > 1 && (this.list[index].proprice = (this.list[index].proprice - price).
    toFixed(2))
  },
},
```

2.3_组件通信_子向父_自定义事件

目标：子组件触发父自定义事件方法

2. 子组件内, 恰当的时机, 触发父给我绑的自定义事件, 导致父methods里事件处理函数执行

```
<template>
  <div class="my-product">
    <h3>标题: {{ title }}</h3>
    <p>价格: {{ price }}元</p>
    <p>{{ info }}</p>
    <p>
      <button @click="kanFn">砍价</button>
    </p>
  </div>
</template>
<script>
export default {
  props: ['index', 'title', 'price', 'info'],
  methods: {
    kanFn(){
      this.$emit('subprice', this.index, 1)
    }
  }
}
</script>
```

什么时候使用子传父技术?

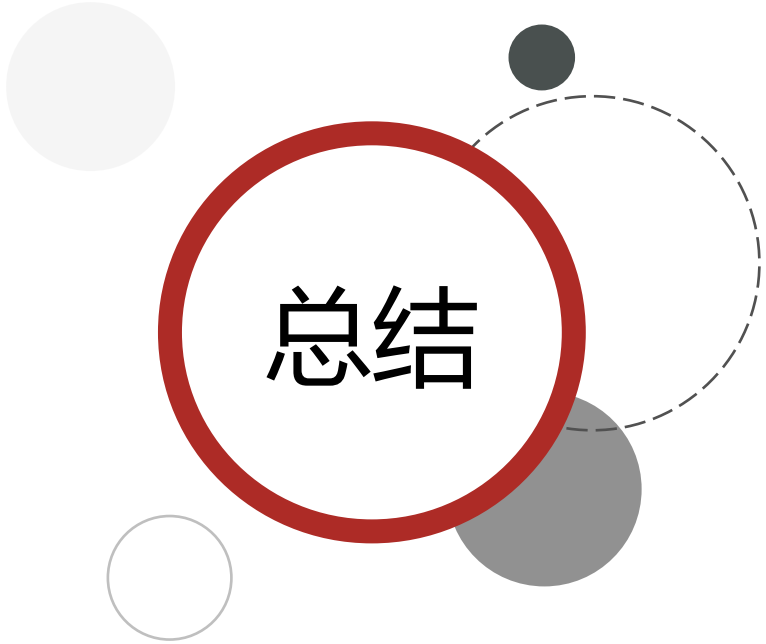
当子想要去改变父里的数据

子传父如何实现?

父组件内, 给组件@自定义事件="父methods函数"

子组件内, 恰当时机this.$emit('自定义事件名', 值)

2.4_阶段总结



组件是什么?

是一个vue实例, 封装标签, 样式和JS代码

组件好处?

便于复用, 易于扩展

组件通信哪几种, 具体如何实现?

父 -> 子

父 <- 子

问题: 两个没有任何引入关系的组件, 要如何互相通信呢?

2.5_组件通信-EventBus

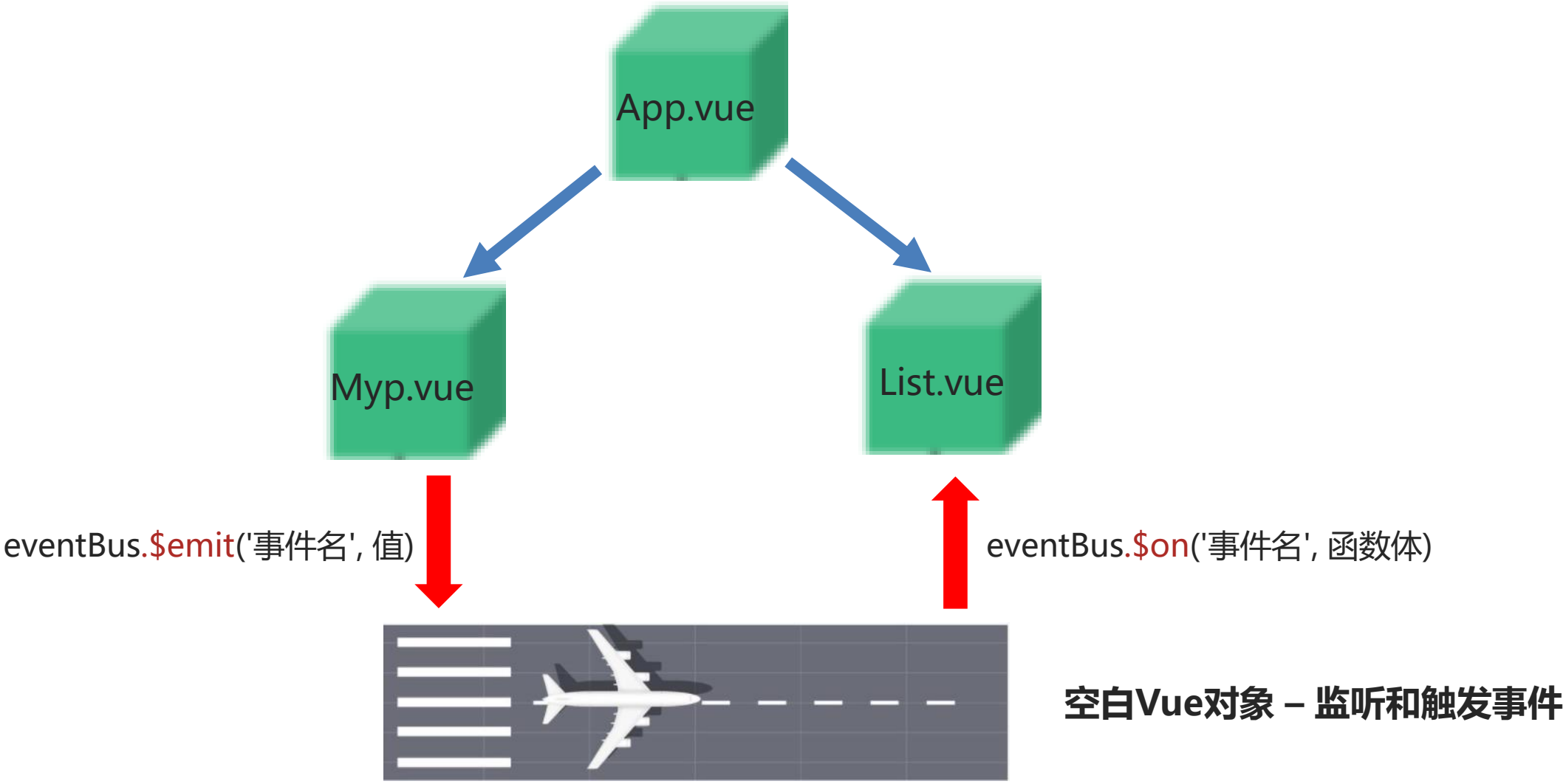
目标：App.vue里引入MyProduct.vue和List.vue

MyProduct.vue

List.vue

2.5_组件通信-EventBus

目标：常用于跨组件通信时使用



2.5_组件通信-EventBus

目标：常用于跨组件通信时使用

- 语法
  - src/EventBus/index.js – 创建空白Vue对象并导出
  - 在要接收值的组件(List.vue)　　eventBus.$on('事件名', 函数体)
  - 在要传递值的组件(MyProduct.vue)　eventBus.$emit('事件名', 值)

```
import eventBus from '../EventBus'
export default {
  props: ['index', 'title', 'price', 'intro'],
  methods: {
    subFn(){
      eventBus.$emit("send", this.index, 1)
    }
  }
}
```

```
import eventBus from '../EventBus'
export default {
  props: ['arr'],
  created(){
      eventBus.$on('send', (index, price) => {
        this.arr[index].proprice -= price
      })
  }
}
```

什么时候使用eventBus技术?

当2个没有引用关系的组件之间要通信传值

eventBus技术本质是什么?

空白Vue对象, 只负责$on和$emit

目录

Contents

Todo案例

3.0_todo案例_创建工程和组件

案例

创建工程和组件

需求1：创建新工程

需求2：分组件创建 – 准备标签和样式(从.md笔记复制)

效果如下：

TodoHeader.vue

TodoMain.vue

TodoFooter.vue

3.0_todo案例_创建工程和组件

案例

创建工程和组件

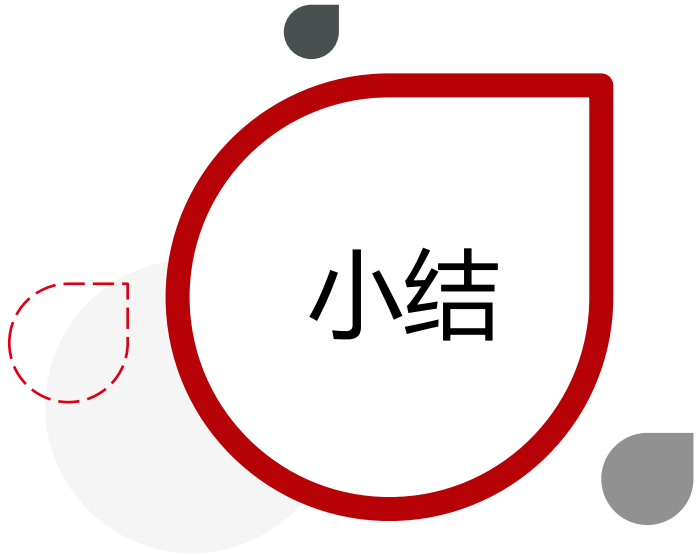
分析：

①：初始化todo工程

②：创建 3 个组件和里面代码(在预习资料.md复制)

③：把styles的样式文件准备好(从预习资料复制)

④： App.vue引入注册使用, 最外层容器类名todoapp



涉及到了哪些技术点?

组件创建

组件引入

组件注册

组件使用

3.1_todo案例_铺设待办任务

案例

循环展示任务

需求1：把待办任务，展示到页面TodoMain.vue组件上

需求2：关联选中状态，设置相关样式

效果如下：

3.1_todo案例_铺设待办任务

循环展示任务

分析：

①：App.vue – 准备数组传入TodoMain.vue内

②：v-for循环展示数据

③：v-model绑定复选框选中状态

④：根据选中状态, 设置完成划线样式

涉及到了哪些技术点?

父传子技术

v-for循环

v-model绑定

动态class使用

3.2_todo案例_添加任务

案例

添加功能

需求：输入任务敲击回车，新增待办任务

效果如下：

3.2_todo案例_添加任务

案例

添加功能

分析：

①：TodoHeader.vue – 输入框 – 键盘事件 – 回车按键

②：子传父, 把待办任务 – App.vue中 – 加入数组list里

③：原数组改变, 所有用到的地方都会更新

④：输入框为空, 提示用户必须输入内容

涉及到了哪些技术点?

键盘事件, enter修饰符

子传父技术

空值判断技术

3.3_todo案例_删除任务

案例

删除功能

需求：点击任务后的x, 删除当前这条任务

效果如下：

3.3_todo案例_删除任务

案例

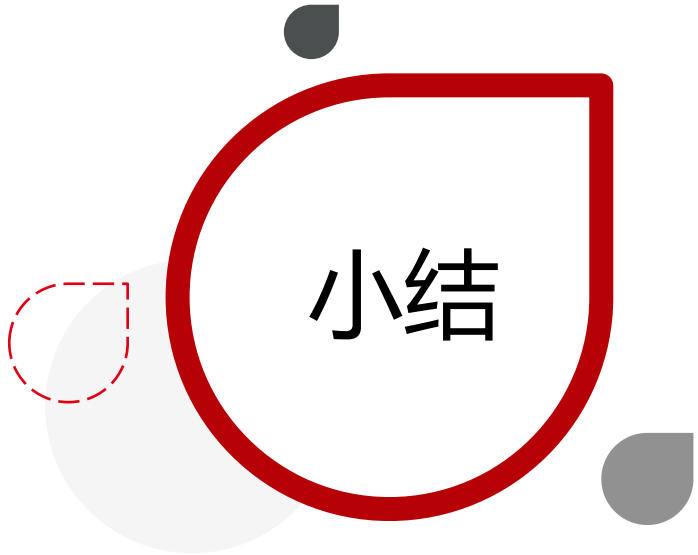
删除功能

分析：

①：x标签 – 点击事件 – 传入id区分

②：子传父, 把id传回– App.vue中 – 删除数组list里某个对应的对象

③：原数组改变, 所有用到的地方都会更新



涉及到了哪些技术点?

点击事件, 传id

子传父

数组删除某个元素, v-for更新

3.4_todo案例_底部统计

案例

底部统计

需求：统计当前任务的条数

效果如下：

案例

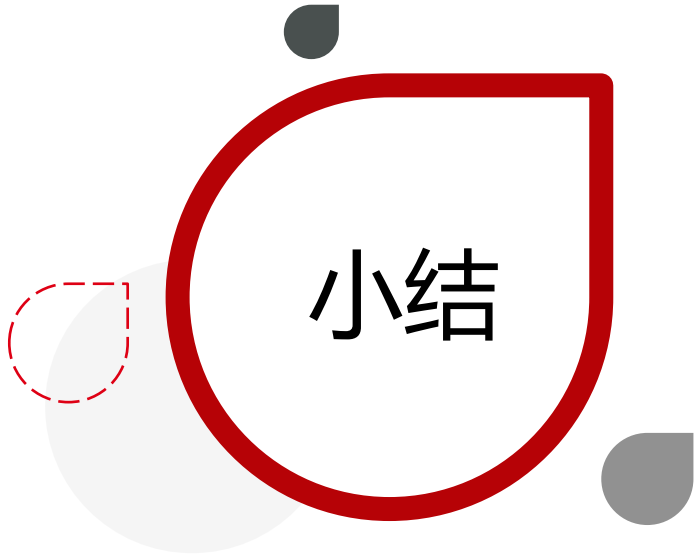
底部统计

分析：

①：App.vue中 – 数组list – 传给TodoFooter.vue

②：直接在标签上显示 / 定义计算属性用于显示都可以

③：原数组只要改变, 所有用到此数组的地方都会更新



涉及到了哪些技术点?

父传子

计算属性

数组更新 – 所有地方受影响

3.5_todo案例_数据切换

案例

数据切换

需求1：点击底部切换 – 点谁谁有边框

需求2：对应切换不同数据显示

效果如下：

3.5_todo案例_数据切换

案例

数据切换

分析：

①：TodoFooter.vue – 定义isSel – 值为all, yes, no其中一种

②：多个class分别判断谁应该有类名selected

③：点击修改isSel的值

④：子传父, 把类型isSel传到App.vue

⑤：定义计算属性showArr, 决定从list里显示哪些数据给TodoMain.vue和TodoFooter.vue

涉及到了哪些技术点?

动态class – 配合判断

子传父技术

计算属性+数组过滤方法

3.6_todo案例_清空已完成

案例

清空已完成

需求：点击右下角链接标签, 清除已完成任务

效果如下:

3.6_todo案例_清空已完成

案例

清空已完成

分析：

①：清空标签 – 点击事件

②：子传父 – App.vue – 一个清空方法

③：过滤未完成的覆盖list数组 (不考虑恢复)

涉及到了哪些技术点?

子传父技术

数组过滤方法

3.7_todo案例_数据缓存

案例

数据缓存

需求：无论如何变化 – 都保证刷新后数据还在

效果如下：

案例

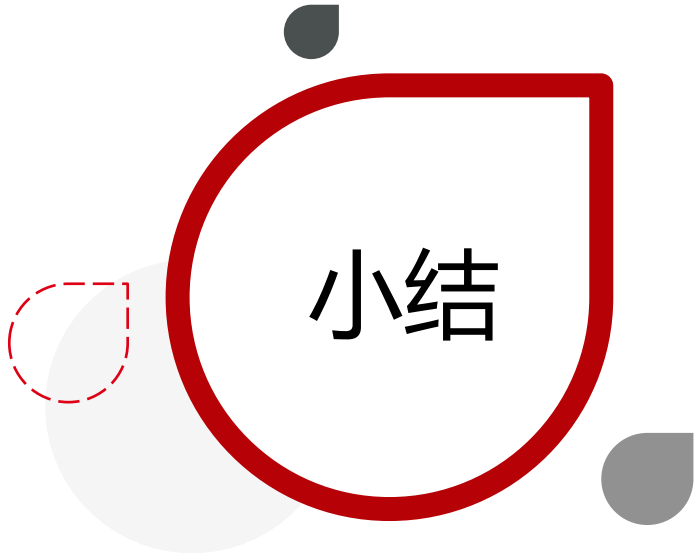
数据缓存

分析：

①：App.vue – 侦听list数组改变 – 深度

②：覆盖式存入到本地 – 注意本地只能存入JSON字符串

③：刷新页面 – list应该默认从本地取值 – 要考虑无数据情况空数组



涉及到了哪些技术点?

深度侦听

数据缓存

序列化和反序列化

3.8_todo案例_全选功能

案例

全选功能

需求1：点击全选 – 小选框受到影响

需求2：小选框都选中(手选) – 全选自动选中状态

效果如下：

3.8_todo案例_全选功能

案例

全选功能

分析：

①：TodoHeader.vue – 计算属性 - isAll

②：App.vue – 传入数组list – 在isAll的set里影响小选框

③：isAll的get里统计小选框最后状态, 影响isAll – 影响全选状态

④：考虑无数据情况空数组 – 全选不应该勾选

涉及到了哪些技术点?

计算属性完整写法

every方法如果不遍历返回true

今日总结

1. 什么是组件
2. 组件使用步骤
3. 什么是父子组件关系
4. 组件通信: 父 => 子
5. 组件通信: 父 <= 子
6. 组件通信: 兄弟传值方式

附加练习

1. 今天案例比较大, 练习较多, 案例可以跟着视频做
2. 在md文档最后还有附加练习, 然后再挑战作业

今日作业

1. 在md文件最后, 有2个作业 – 挑战一下吧

传智教育旗下高端IT教育品牌