vue基础 - day03

组件基础 + 组件通讯

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

1. 掌握 组件基础，完成组件定义注册和使用
2. 掌握 组件通讯，完成父传子和子传父数据

目录

Contents

案例效果：

目录

Contents

问题1：以前遇到重复的结构代码，怎么做的？

问题2：复制粘贴？可维护性高吗？

为什么使用组件?


案例：折叠面板
芙蓉楼送辛渐
收起
寒雨连江夜入吴，
平明送客楚山孤。
洛阳亲友如相问，
一片冰心在玉壶。
<template>

<h3>案例：折叠面板</h3>

<h4>芙蓉楼送辛渐</h4>
<span class="btn" @click="isShow = !isShow">
{{ isShow ? '收起' : '展开' }}

寒雨连江夜入吴，
平明送客楚山孤。
洛阳亲友如相问，
一片冰心在玉壶。

</template>


案例：折叠面板
芙蓉楼送辛渐
收起
寒雨连江夜入吴，
平明送客楚山孤。
洛阳亲友如相问，
一片冰心在玉壶。
芙蓉楼送辛渐
收起
寒雨连江夜入吴，
平明送客楚山孤。
洛阳亲友如相问，
一片冰心在玉壶。
芙蓉楼送辛渐
收起
寒雨连江夜入吴，
平明送客楚山孤。
洛阳亲友如相问，
一片冰心在玉壶。

为什么使用组件?

组件的好处: 各自独立, 便于复用

```
<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>
```

小结

1. 遇到重复标签结构想复用?

封装成组件

2. 封装组件的好处?

各自独立, 便于复用，可维护性高

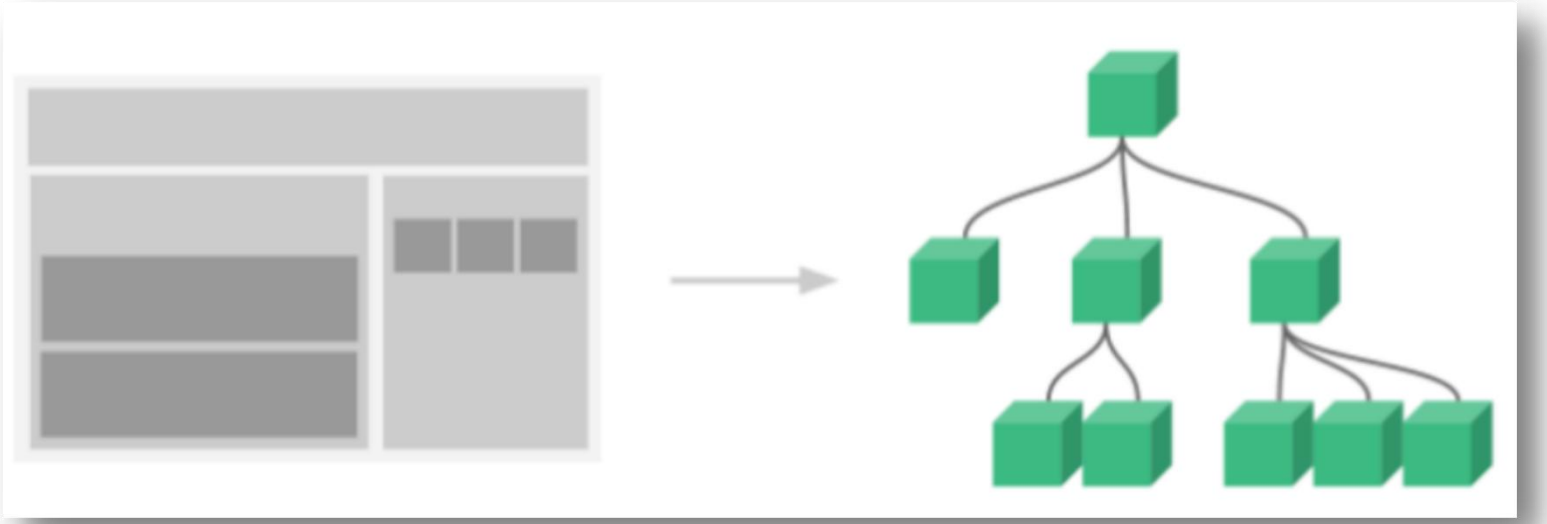
什么是组件化开发？

什么是组件化开发

- 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)
- 例如：http://www.ibootstrap.cn/ 所展示的效果，就契合了组件化开发的思想。

前端组件化开发的好处：

- 提高了 复用性和灵活性
- 提升了 开发效率 和 后期可维护性



小结

什么是组件?

可复用的vue实例, 封装html结构, 样式, JS

什么时候封装组件?

遇到重复标签, 可复用的时候

组件化好处?

复用性和灵活性

开发效率 和 后期可维护性

如何 创建 和 使用 组件?

组件的注册使用

App.vue 是根组件, 这个比较特殊, 是最大的一个根组件。其实里面还可以注册使用其他小组件。

使用组件的四步：

1. 创建组件, 封装要复用的标签, 样式, JS代码
2. 引入组件
3. 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
4. 使用组件：组件名当成标签使用即可

   <组件名></组件名>

注意点：组件名不能和内置的html名同名

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

Vue.component("组件名", 组件对象)
```

```
import 组件对象 from 'vue文件路径'

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

小结

创建和使用组件步骤?

1. 创建 .vue文件 - 标签 - 样式 - JS
2. 引入 组件
3. 注册 组件 (全局 / 局部)
4. 使用 组件 (组件名用作标签)

组件的名字，有没有什么规范和要求呢？

组件名的大小写

在进行组件的注册时，定义组件名的方式有两种：

1. 注册使用短横线命名法，例如 hm-header 和 hm-main

```
Vue.component('hm-button', HmButton)
```

使用时 `<hm-button></hm-button>`

2. 注册使用大驼峰命名法，例如 HmHeader 和 HmMain

```
Vue.component('HmButton', HmButton)
```

使用时 `<HmButton></HmButton>` 和 `<hm-button></hm-button>` 都可以

推荐1: 定义组件名时, 用大驼峰命名法, 更加方便

推荐2: 使用组件时，遵循html5规范, 小写横杠隔开（可选）

通过 name 注册组件 (了解)

组件在 开发者工具中 显示的名字，可以通过name进行修改：

```vue
<template>
  <button>按钮组件</button>
</template>

<script>
export default {
  name: 'HmButton'
}
</script>

<style lang="less">
button {
  width: 80px;
  height: 50px;
  border-radius: 5px;
  background-color: pink;
}
</style>
```

在注册组件期间，除了可以直接提供组件的注册名称之外，

还可以把组件的 name 属性作为注册后组件的名称

```vue
import HmButton from './components/HmButton/index.vue'
Vue.component(HmButton.name, HmButton)
// 等价于 Vue.component('HmButton', HmButton)
```

组件与组件之间的样式会互相影响么？

组件的样式冲突 scoped

默认情况下，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。

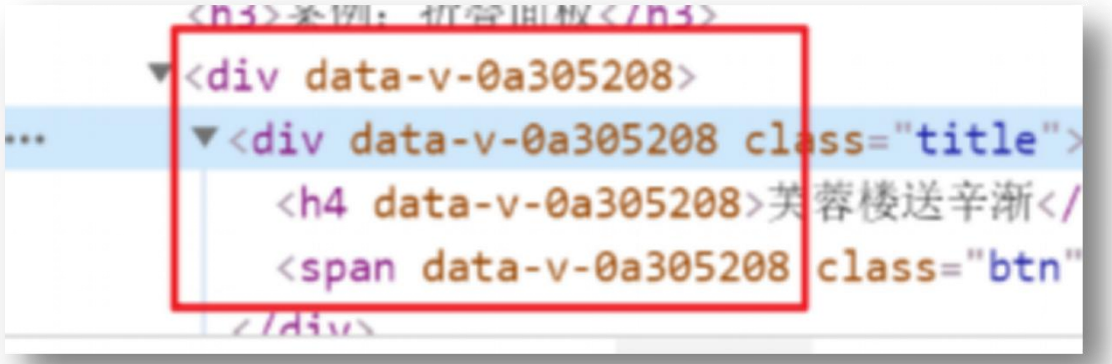
1. `全局样式`: 默认组件中的样式会作用到全局

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

scoped原理?

- （1）当前组件内标签都被添加 data-v-hash值 的属性
- （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, 才会被这个样式作用到



小结

希望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？

（1）会自动给 组件内标签 添加data-v-hash值属性

（2）所有选择器都带上属性选择器

目录

Contents

每个组件有自己的数据, 提供在data中

每个组件数据独立, 组件数据无法互相直接访问 (合理的)

但是如果需要跨组件访问数据，怎么办呢？ => 组件通信

组件通信的方式有很多: 现在先关注两种，父传子 子传父

组件通信 - 父传子

需求：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如图所示

父传子语法:

1. 父组件通过给子组件加属性传值

```
<Son price="100" title="不错" :info="msg"></Son>
```

2. 子组件中, 通过props属性接收

```
props: ['price', 'title', 'info']
```

利用父传子，可以往子组件传递数据。

父传子的基本步骤是什么?

父组件内，给子组件添加属性的方式传值

子组件内，通过 props 接收

以后的数据，肯定是后台回来的，商品应该是动态渲染的

问题1: 一个个的写组件合理么？能循环使用组件吗?

问题2: 循环使用组件, 又如何向组件内传值呢?

v-for 遍历展示组件练习

需求: 遍历展示商品列表

1. 准备数据：假定, 发送请求回来的商品数据,

```
list: [
  { id: 1, proname: '超级好吃的棒棒糖', proprice: 18.8 },
  { id: 2, proname: '超级好吃的大鸡腿', proprice: 34.2 },
  { id: 3, proname: '超级无敌的冰激凌', proprice: 14.2 }
]
```

2. v-for 遍历展示

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

小结

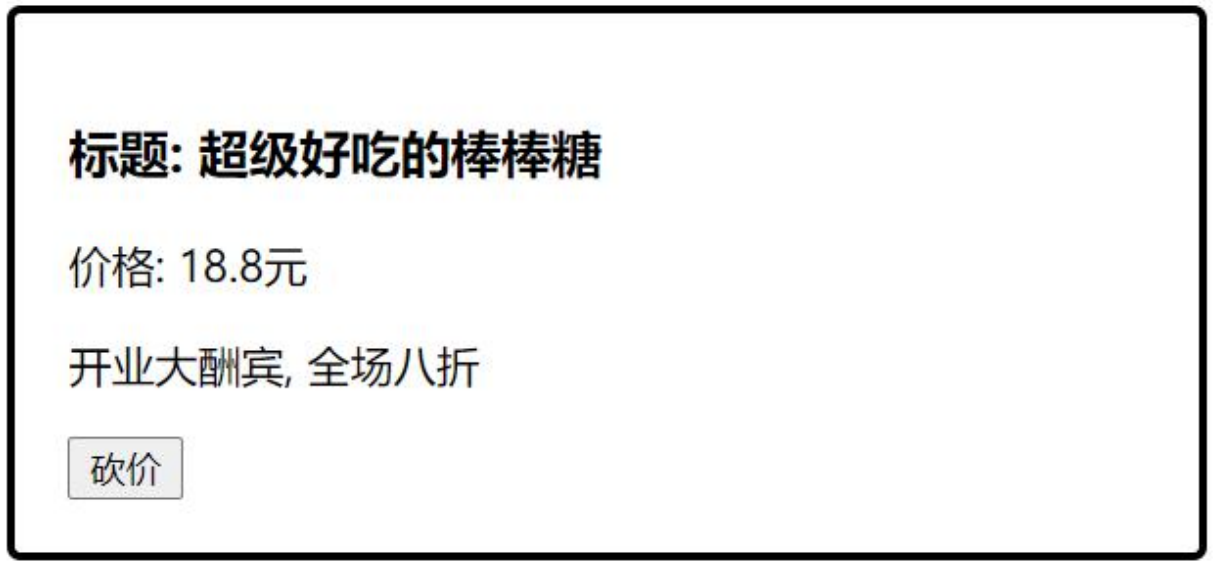
组件可以循环遍历显示么？

配合 v-for 即可

思考

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：子组件内能直接改变，父传入的数据吗？



单向数据流

在vue中需要遵循单向数据流原则： （从父到子的单向数据流动，叫单向数据流）

1. 父组件的数据变化了，会自动向下流动影响到子组件

2. 子组件不能直接修改父组件传递过来的 props， props是只读的！

```
▶ [Vue warn]: Avoid mutating a prop directly since the value
renders. Instead, use a data or computed property based on th

found in

---> <Product> at src/components/MyProduct.vue
       <App> at src/App.vue
         <Root>
```

小结

什么是单向数据流?

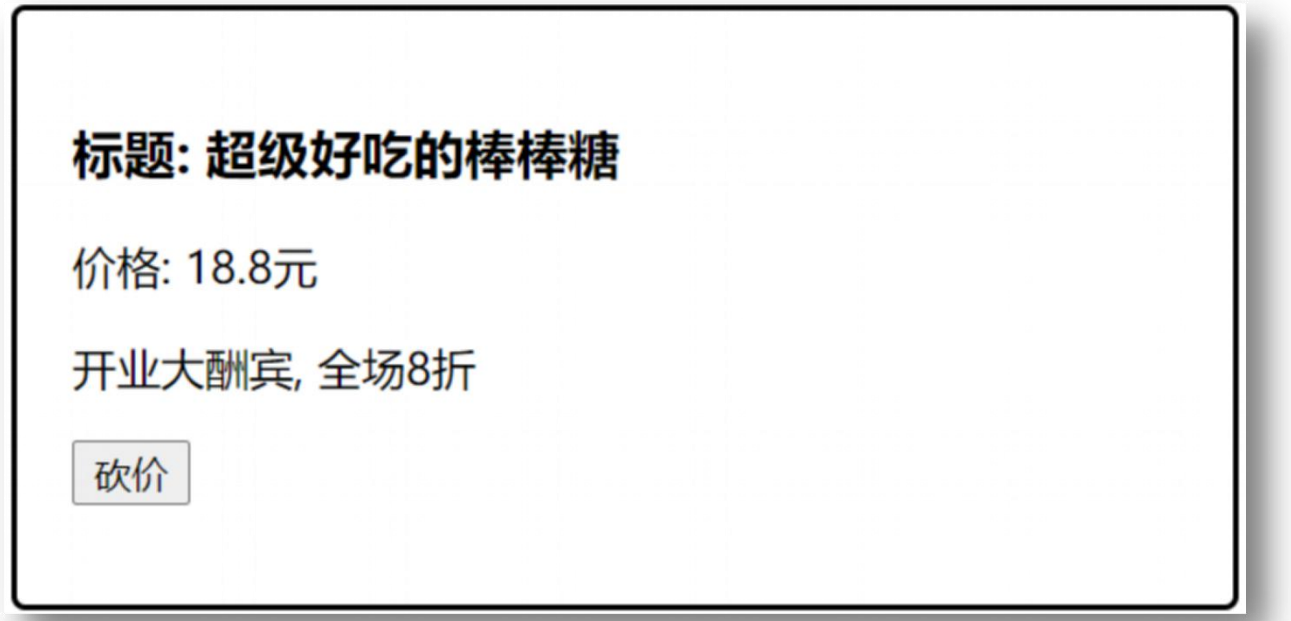
从父到子的单向数据流动, 叫单向数据流

props里定义的变量能修改吗?

不能, props里的变量本身是只读的

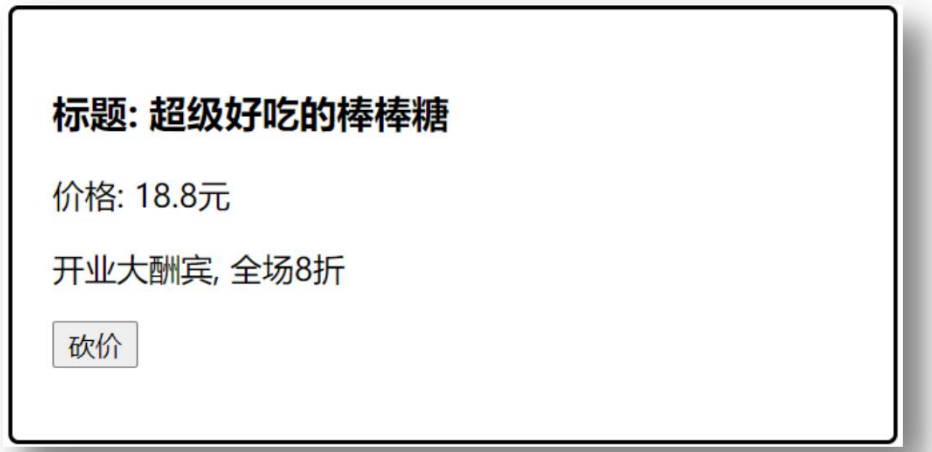
思考

问题：那子组件如何才能修改父组件里的数据呢？



组件通信 - 子传父

- 需求: 商品组件, 实现砍价功能



- 前置补充, 父 -> id -> 子组件 (用于区分哪个子组件)

```
<Product
  v-for="item in list"
  :key="item.id"
  :title="item.proname"
  :price="item.proprice"
  :info="str"
  :proId="item.id">
</Product>
```

组件通信 - 子传父

子传父的基本语法：

1. 子组件可以通过 `this.$emit('事件名', 参数1, 参数2, ...)` 触发事件的同时传参的

```
this.$emit('sayPrice', 2)
```

2. 父组件可以给子组件注册对应的自定义事件

```
<MyProduct
  ...
  @sayPrice="sayPrice">
</MyProduct>
```

父组件并提供对应的函数接收参数

```
methods: {
  sayPrice (num) {
    console.log(num)
  }
},
```

小结

利用子传父，可以让子组件调用到父组件的方法修改父的数据

子传父的步骤是什么呢?

1. 子组件内, 恰当时机 this.$emit('自定义事件名', 值)

2. 父组件内, 给组件 @自定义事件="父methods函数"

props 是父传子, 传递给子组件的数据，直接决定了子组件的展示

props传值可以乱传吗？ 万一不小心传错类型了呢？

props校验

说明：props 是父传子，传递给子组件的数据，为了提高 子组件被使用时 的稳定性，可以进行props校验，验证传递的数据是否符合要求

默认的数组形式，不会进行校验，如果希望校验，需要提供对象形式的 props

```
props: {
    ...
}
```

官网地址：官方文档

props 提供了多种数据验证方案，例如：

- 基础的类型检查 Number
- 多个可能的类型 [String, Number]
- 必填项校验 required: true
- 默认值 default: 100
- 自定义验证函数

小结

为了提高 子组件被使用时 的稳定性，防止props传值类型等传错，可以怎么办？

配置props校验

目录

Contents

案例

创建工程和组件

需求1：准备标签和样式 （教学资料中拷贝）

需求2：拆分组件局部注册

效果如下：

TodoHeader.vue

TodoMain.vue

TodoFooter.vue

要做列表渲染，数据应该放在哪个组件中呢？

案例

列表渲染

效果如下：

注意：公共的数据应该放在父组件 App.vue 中！

案例

列表渲染

步骤分析：

①：在`App.vue`提供任务列表数据

②：App.vue通过父传子，把list数据传给 `TodoMain.vue`

③：`TodoMain.vue`接受数据，且渲染

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：

案例

删除功能

步骤分析：

①：x标签 - 点击事件 - 传入id区分

②：子传父，把id传回- App.vue中

③：删除数组list里某个对应的对象

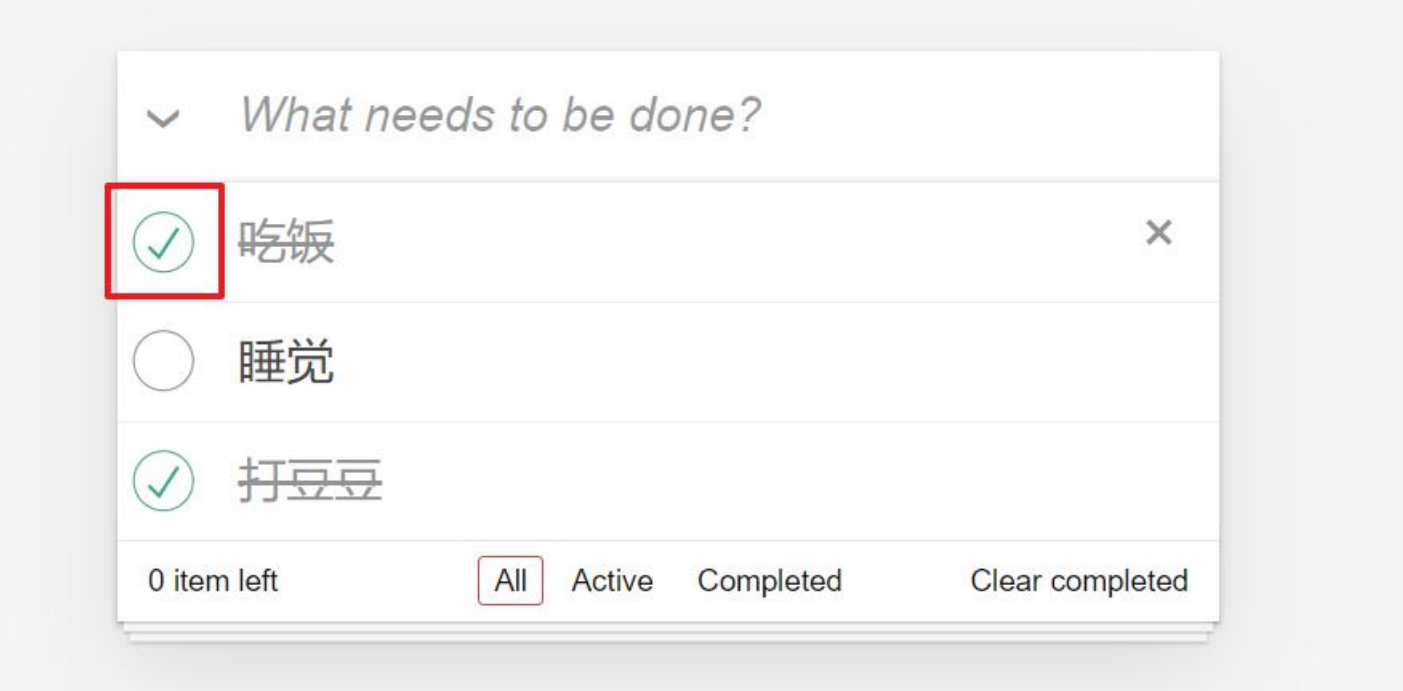
原数组改变，所有用到的地方都会更新

案例

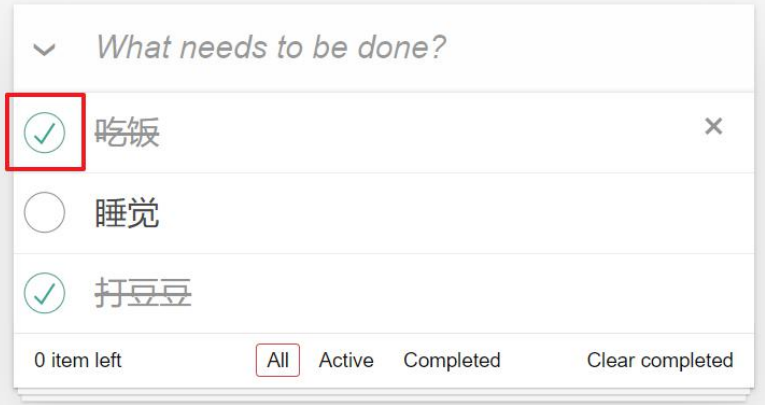
修改状态

需求：点击左侧的任务状态

修改状态：



案例 修改状态



注意：要把`v-model` 改成 `:checked`

原因：`v-model`和父组件双向数据绑定，违反单向数据流的原则。

步骤分析：

①：把`v-model` 改成 `:checked`

②：给checkbox注册change事件， 子传父，把id传回- App.vue中

③：父组件注册事件，修改状态

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：

案例 **添加功能**

步骤分析：

①：在`TodoHeader.vue`组件中通过v-model获取到任务的名字

②：回车的时候，需要子传父，把名字传给父组件

③：父组件接受name，并且进行添加，数据变了，所有地方自动更新

④：输入框为空，提示用户必须输入内容

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：定义 计算属性 用于显示底部未完成任务数

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：

案例

清空已完成

分析：

①：提供计算属性，用于控制清空按钮的显示和隐藏

②：通过v-show控制显示隐藏，注册了点击事件

③：触发事件子传父

④：父组件过滤清空已经完成的任务

案例

底部筛选功能

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：

案例

底部筛选功能

分析：

①：TodoFooter.vue - 给3个a注册点击事件

②：准备 type 数据，记录点击的按钮 （all active completed）

③：动态控制 selected类名

④：子传父, 把类型 type 传到App.vue

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

案例

本地存储

需求：无论如何变化 - 都保证刷新后数据还在

效果如下：

案例

本地存储

步骤分析：

①：App.vue - 侦听list数组是否改变 - 深度

②：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

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：

案例

全选功能

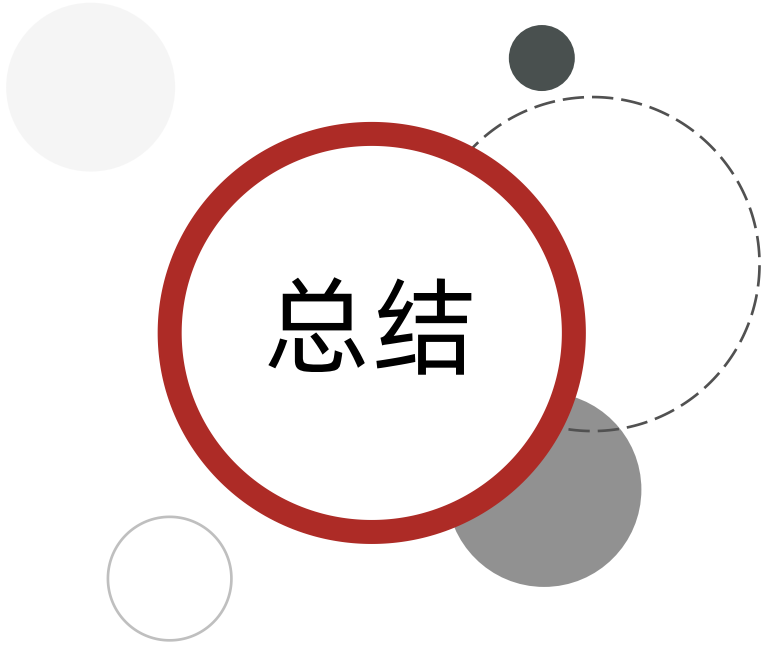
分析：

①：`TodoMain.vue` - 计算属性 - isCheckAll

②：计算属性的完整写法 get set

③：set 时，子传父通知到父组件更新

④：父组件接受到参数，更新同步



1. 什么是组件
2. 组件使用步骤
3. 组件通信: 父 => 子
4. 组件通信: 父 <= 子

把课上例子，练习，案例做一遍

预习第二天内容

传智教育旗下高端IT教育品牌