

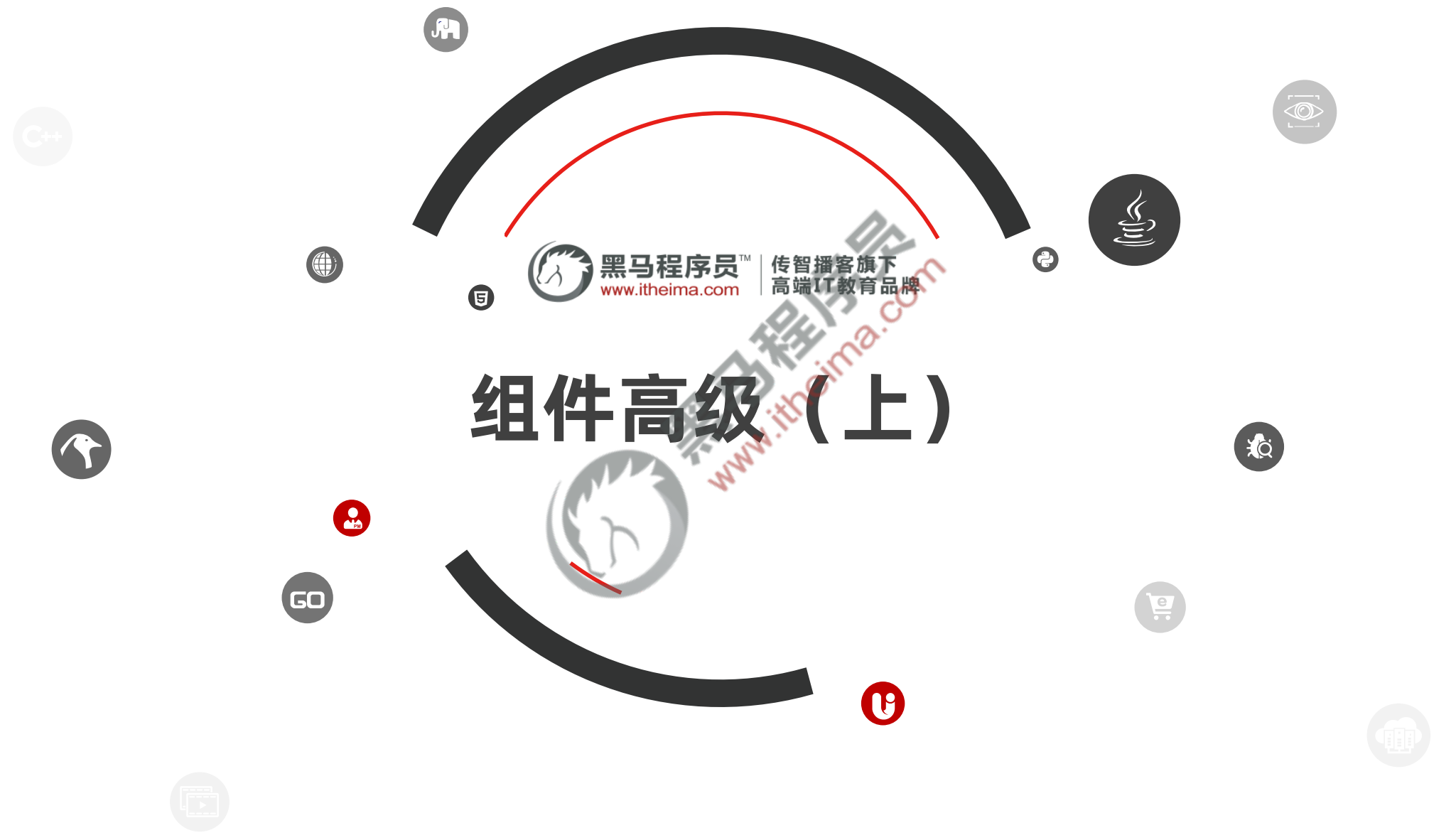


黑马程序员™
www.itheima.com

传智播客旗下
高端IT教育品牌



组件高级（上）



录 Contents

- ◆ watch 侦听器
- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ vue 3.x 中全局配置 axios
- ◆ 购物车案例

1. 什么是 watch 侦听器

watch 侦听器允许开发者监视数据的变化，从而**针对数据的变化做特定的操作**。例如，监视用户名的变化并发起请求，判断用户名是否可用。



黑马程序员
www.itheima.com

2. watch 侦听器的基本语法

开发者需要在 **watch 节点** 下，定义自己的侦听器。实例代码如下：

```
1 export default {  
2   data() {  
3     return { username: '' }  
4   },  
5   watch: {  
6     // 监听 username 的值的变化，  
7     // 形参列表中，第一个值是“变化后的新值”，第二个值是“变化之前的旧值”  
8     username(newVal, oldVal) {  
9       console.log(newVal, oldVal)  
10    },  
11  },  
12 }
```

3. 使用 watch 检测用户名是否可用

监听 username 值的变化，并使用 axios 发起 Ajax 请求，检测当前输入的用户名是否可用：

```
1 import axios from 'axios'
2 export default {
3   data() {
4     return { username: '' }
5   },
6   watch: {
7     async username(newVal, oldVal) {
8       const { data: res } = await axios.get(`https://www.escook.cn/api/finduser/${newVal}`)
9       console.log(res)
10    },
11  },
12 }
```

4. immediate 选项

默认情况下，组件在初次加载完毕后不会调用 watch 侦听器。如果能让 watch 侦听器立即被调用，则需要使用 **immediate** 选项。实例代码如下：

```
1 watch: {
2   // 1. 监听 username 值的变化
3   username: {
4     // 2. handler 属性是固定写法：当 username 变化是，调用 handler
5     async handler(newVal, oldVal) {
6       const { data: res } = await axios.get(`https://www.escook.cn/api/finduser/${newVal}`)
7       console.log(res)
8     },
9     // 3. 表示组件加载完毕后立即调用一次当前的 watch 侦听器
10    immediate: true
11  },
12 },
```

5. deep 选项

当 watch 侦听的是一个对象，如果对象中的属性值发生了变化，则无法被监听到。此时需要使用 deep 选项，代码示例如下：

```
1 data() {
2   return {
3     info: { username: 'admin' }, // info 中包含 username 属性
4   }
5 },
6 watch: {
7   info: { // 直接监听 info 对象的变化
8     async handler(newVal, oldVal) {
9       const { data: res } = await axios.get(`https://www.escook.cn/api/finduser/${newVal.username}`)
10      console.log(res)
11    },
12    deep: true // 需要使用 deep 选项，否则 username 值的变化无法被监听到
13  },
14 },
```



6. 监听对象单个属性的变化

如果只想监听对象中单个属性的变化，则可以按照如下的方式定义 watch 侦听器：

```
1 data() {
2   return {
3     info: { username: 'admin', password: '' }, // info 中包含 username 属性
4   }
5 },
6 watch: {
7   'info.username': { // 只想监听 info.username 属性值的变化
8     async handler(newVal, oldVal) {
9       const { data: res } = await axios.get(`https://www.escook.cn/api/finduser/${newVal}`)
10      console.log(res)
11    },
12  },
13 },
```


7. 计算属性 vs 侦听器

计算属性和侦听器侧重的应用场景不同：

计算属性侧重于监听多个值的变化，最终计算并返回一个新值

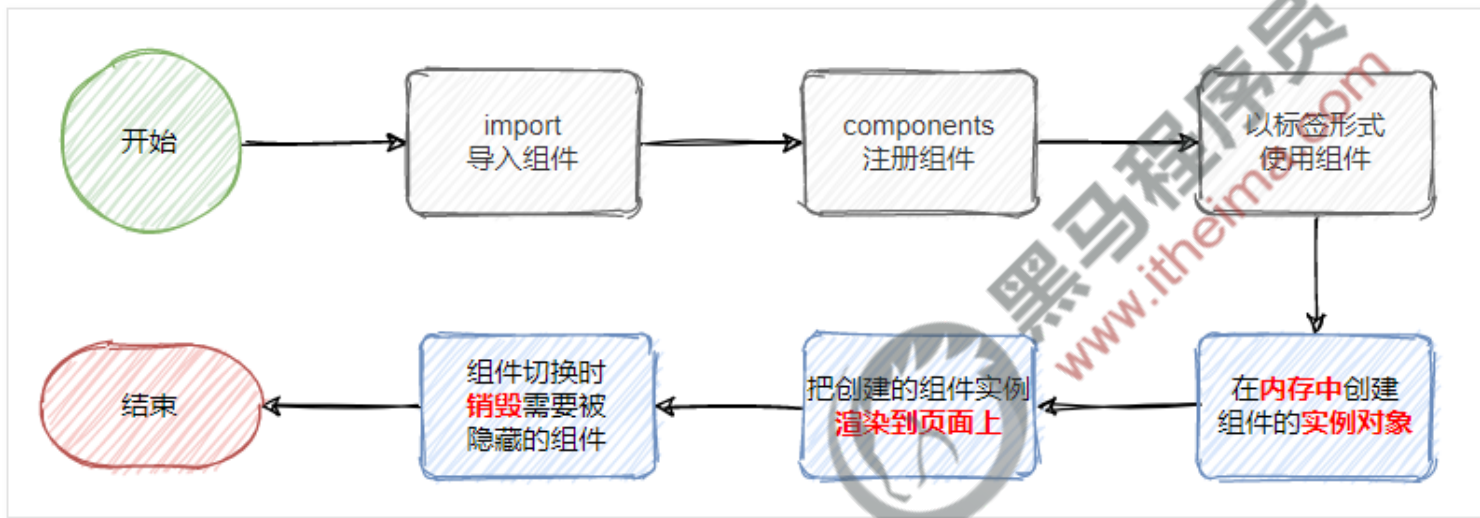
侦听器侧重于监听单个数据的变化，最终执行特定的业务处理，不需要有任何返回值



录 Contents

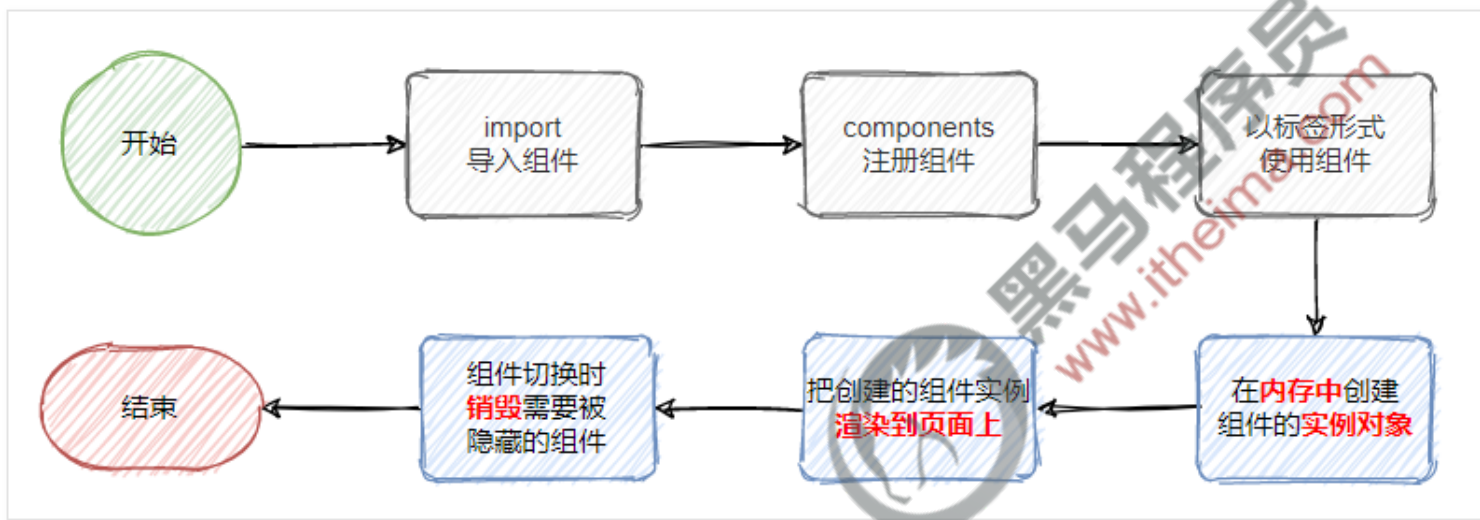
- ◆ watch 侦听器
- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ vue 3.x 中全局配置 axios
- ◆ 购物车案例

1. 组件运行的过程



组件的生命周期指的是：组件从**创建** -> **运行**（渲染） -> **销毁**的整个过程，强调的是**一个时间段**。

2. 如何监听组件的不同时刻



vue 框架为组件内置了不同时刻的生命周期函数，生命周期函数会伴随着组件的运行而自动调用。例如：

- ① 当组件在内存中被创建完毕之后，会自动调用 `created` 函数
- ② 当组件被成功的渲染到页面上之后，会自动调用 `mounted` 函数
- ③ 当组件被销毁完毕之后，会自动调用 `unmounted` 函数

3. 如何监听组件的更新

当组件的 **data 数据更新** 之后，vue 会 **自动重新渲染组件** 的 DOM 结构，从而保证 **View 视图** 展示的数据和 **Model 数据源** 保持一致。

当组件被 **重新渲染完毕** 之后，会自动调用 **updated** 生命周期函数。



4. 组件中主要的生命周期函数

生命周期函数	执行时机	所属阶段	执行次数	应用场景
created	组件在内存中创建完毕后	创建阶段	唯一1次	发 ajax 请求初始数据
mounted	组件初次在页面中渲染完毕后	创建阶段	唯一1次	操作 DOM 元素
updated	组件在页面中被重新渲染完毕后	运行阶段	0 或 多次	-
unmounted	组件被销毁后（页面和内存）	销毁阶段	唯一1次	-

注意：在实际开发中，**created** 是最常用的生命周期函数！

5. 组件中全部的生命周期函数

生命周期函数	执行时机	所属阶段	执行次数	应用场景
beforeCreate	在内存中开始创建组件之前	创建阶段	唯一1次	-
created	组件在内存中创建完毕后	创建阶段	唯一1次	发 ajax 请求初始数据
beforeMount	在把组件初次渲染到页面之前	创建阶段	唯一1次	-
mounted	组件初次在页面中渲染完毕后	创建阶段	唯一1次	操作 DOM 元素
beforeUpdate	在组件被重新渲染之前	运行阶段	0 或 多次	-
updated	组件在页面中被重新渲染完毕后	运行阶段	0 或 多次	-
beforeUnmount	在组件被销毁之前	销毁阶段	唯一1次	-
unmounted	组件被销毁后（页面和内存）	销毁阶段	唯一1次	-

疑问：为什么不在 beforeCreate 中发 ajax 请求初始数据？

6. 完整的生命周期图示

可以参考 vue 官方文档给出的“**生命周期图示**”，进一步理解组件生命周期执行的过程：

<https://www.vue3js.cn/docs/zh/guide/instance.html#生命周期图示>



录 Contents

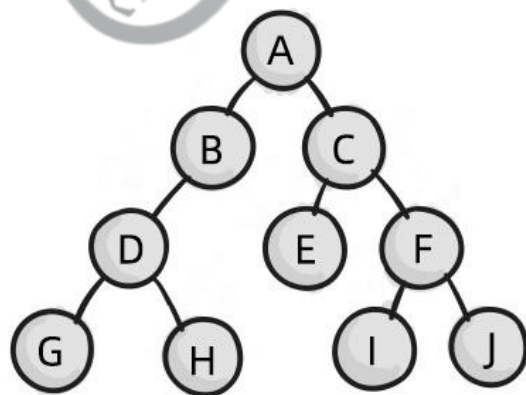
- ◆ watch 侦听器
- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ vue 3.x 中全局配置 axios
- ◆ 购物车案例



1. 组件之间的关系

在项目开发中，组件之间的关系分为如下 3 种：

- ① 父子关系
- ② 兄弟关系
- ③ 后代关系





组件之间的数据共享



黑马程序员
www.itheima.com

传智播客旗下高端IT教育品牌

2. 父子组件之间的数据共享

父子组件之间的数据共享又分为：

- ① 父 -> 子共享数据
- ② 子 -> 父共享数据
- ③ 父 <-> 子双向数据同步



黑马程序员
www.itheima.com



组件之间的数据共享



黑马程序员
www.itheima.com

传智播客旗下高端IT教育品牌

2.1 父组件向子组件共享数据

父组件通过 **v-bind** 属性绑定向子组件共享数据。同时，子组件需要使用 **props** 接收数据。示例代码如下：

```
1 // 父组件
2 <my-test-1 :msg="message"
  :user="userinfo"></my-test-1>
3
4 data() {
5   return {
6     message: 'hello vue',
7     userinfo: { name: 'zs', age: 20 },
8   }
9 }
```

```
1 // 子组件
2 <template>
3   <h3>测试父子传值</h3>
4   <p>{{msg}}</p>
5   <p>{{user}}</p>
6 </template>
7
8 <script>
9   export default {
10     props: ['msg', 'user'],
11   }
12 </script>
```



2.2 子组件向父组件共享数据

子组件通过**自定义事件**的方式向父组件共享数据。示例代码如下：

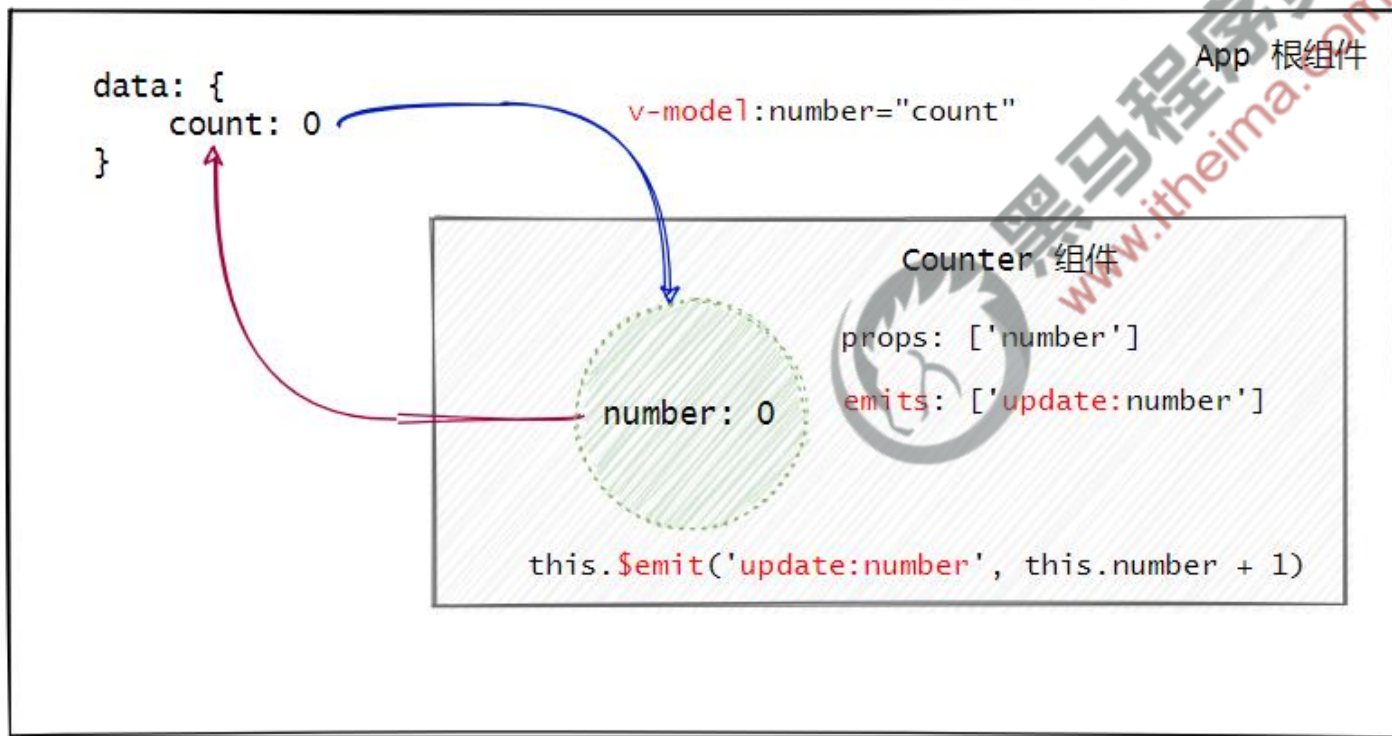
```
1 // 子组件
2 export default {
3   emits: ['n1change'], // 1. 声明自定义事件
4   data() { return { n1: 0 } },
5   methods: {
6     addN1() {
7       this.n1++
8       // 2. 数据变化时，触发自定义的事件
9       this.$emit('n1change', this.n1)
10    },
11  },
12 }
```

```
1 // 父组件
2 // 1. 监听子组件的自定义事件 n1change
3 <my-test-1 @n1change="getn1"></my-test-1>
4
5 export default {
6   data() { return { n1FromSon: 0 } },
7   methods: {
8     getn1(n1) { // 2. 通过形参，接收子组件传递过来的数据
9       this.n1FromSon = n1
10    },
11  },
12 }
```



2.3 父子组件之间数据的双向同步

父组件在使用子组件期间，可以使用 **v-model** 指令维护组件内外数据的双向同步：

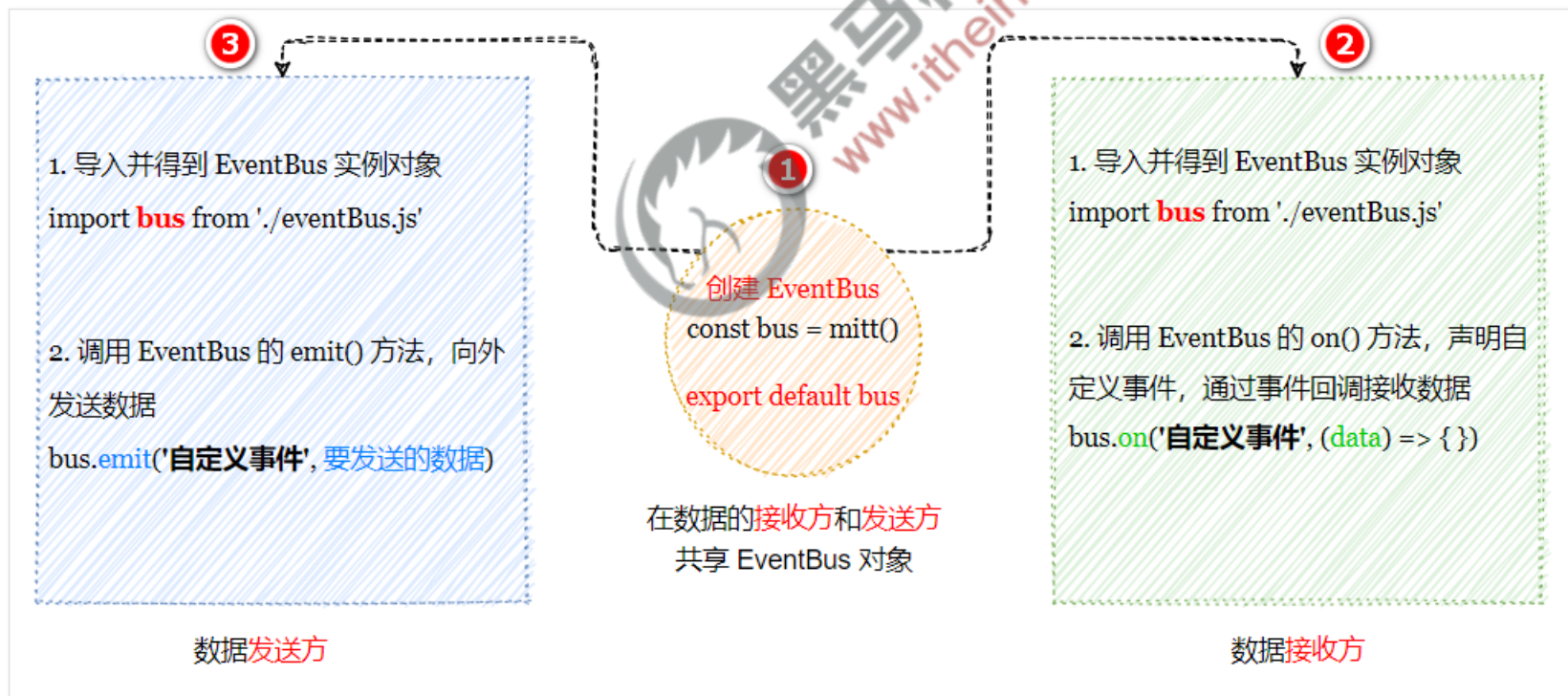




组件之间的数据共享

3. 兄弟组件之间的数据共享

兄弟组件之间实现数据共享的方案是 EventBus。可以借助于第三方的包 mitt 来创建 EventBus 对象，从而实现兄弟组件之间的数据共享。示意图如下：





组件之间的数据共享

3.1 安装 mitt 依赖包

在项目中运行如下的命令，安装 mitt 依赖包：

```
1 npm install mitt@2.1.0
```




组件之间的数据共享



黑马程序员
www.itheima.com

传智播客旗下高端IT教育品牌

3.2 创建公共的 EventBus 模块

在项目中创建公共的 **eventBus** 模块如下：

```
1 // eventBus.js
2
3 // 导入 mitt 包
4 import mitt from 'mitt'
5 // 创建 EventBus 的实例对象
6 const bus = mitt()
7
8 // 将 EventBus 的实例对象共享出去
9 export default bus
```



3.3 在数据接收方自定义事件

在数据接收方，调用 `bus.on('事件名称', 事件处理函数)` 方法注册一个自定义事件。示例代码如下：

```
1 // 导入 EventBus.js 模块，得到共享的 bus 对象
2 import bus from './eventBus.js'
3
4 export default {
5   data() { return { count: 0 } },
6   created() {
7     // 调用 bus.on() 方法注册一个自定义事件，通过事件处理函数的形参接收数据
8     bus.on('countChange', (count) => {
9       this.count = count
10     })
11   }
12 }
```



3.4 在数据接发送方触发事件

在数据发送方，调用 `bus.emit('事件名称', 要发送的数据)` 方法触发自定义事件。示例代码如下：

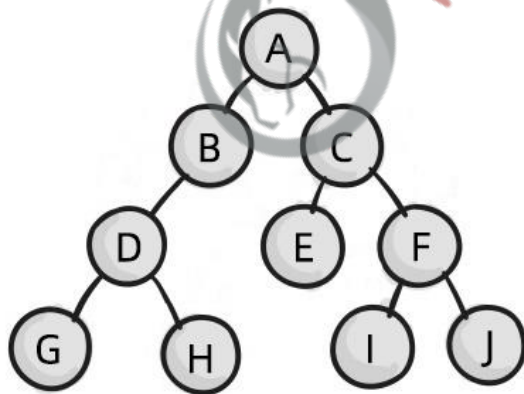
```
1 // 导入 eventBus.js 模块，得到共享的 bus 对象
2 import bus from './eventBus.js'
3
4 export default {
5   data() { return { count: 0 } },
6   methods: {
7     addCount() {
8       this.count++
9       bus.emit('countChange', this.count) // 调用 bus.emit() 方法触发自定义事件，并发送数据
10    }
11  }
12 }
```



4. 后代关系组件之间的数据共享

后代关系组件之间共享数据，指的是父节点的组件向其子孙组件共享数据。此时组件之间的嵌套关系比较复杂，可以使用 `provide` 和 `inject` 实现后代关系组件之间的数据共享。

组件之间的关系





组件之间的数据共享

4.1 父节点通过 **provide** 共享数据

父节点的组件可以通过 **provide** 方法，对其**子孙组件**共享数据：

```
1 export default {
2   data() {
3     return {
4       color: 'red' // 1. 定义"父组件"要向"子孙组件"共享的数据
5     }
6   },
7   provide() { // 2. provide 函数 return 的对象中，包含了"要向子孙组件共享的数据"
8     return {
9       color: this.color,
10    }
11  },
12 }
```



组件之间的数据共享



黑马程序员
www.itheima.com

传智播客旗下高端IT教育品牌

4.2 子孙节点通过 **inject** 接收数据

子孙节点可以使用 **inject** 数组，接收父级节点**向下共享的数据**。示例代码如下：

```
1 <template>
2   <h5>子孙组件 --- {{color}}</h5>
3 </template>
4
5 <script>
6 export default {
7   // 子孙组件，使用 inject 接收父节点向下共享的 color 数据，并在页面上使用
8   inject: ['color'],
9 }
10 </script>
```



4.3 父节点对外共享响应式的数据

父节点使用 provide 向下共享数据时，可以结合 **computed 函数** 向下共享**响应式的数据**。示例代码如下：

```
1 import { computed } from 'vue' // 1. 从 vue 中按需导入 computed 函数
2
3 export default {
4   data() {
5     return { color: 'red' }
6   },
7   provide() {
8     return {
9       // 2. 使用 computed 函数，可以把要共享的数据“包装为”响应式的数据
10      color: computed(() => this.color),
11    }
12  },
13 }
```



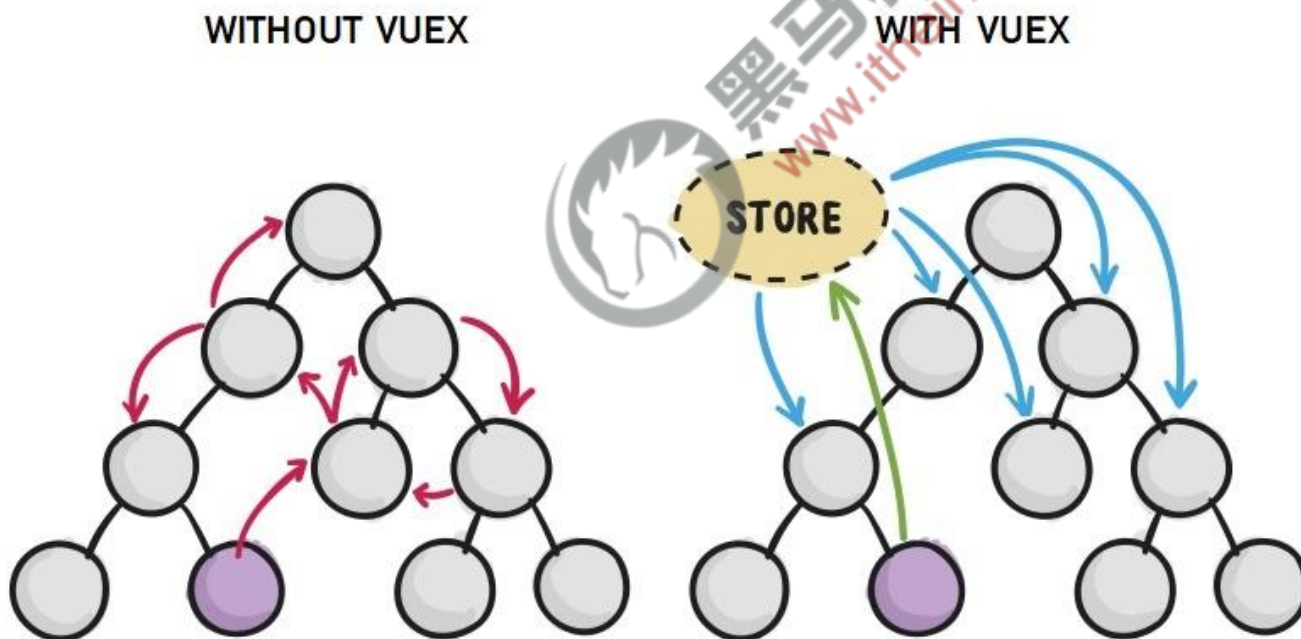
4.4 子孙节点使用响应式的数据

如果父级节点共享的是响应式的数据，则子孙节点必须以 `.value` 的形式进行使用。示例代码如下：

```
1 <template>
2   <!-- 响应式的数据，必须以 .value 的形式进行使用 -->
3   <h5>子孙组件 --- {{color.value}}</h5>
4 </template>
5
6 <script>
7 export default {
8   // 接收父节点向下共享的 color 数据,并在页面上使用
9   inject: ['color'],
10 }
11 </script>
```


5. vuex

vuex 是**终极的**组件之间的数据共享方案。在企业级的 vue 项目开发中, vuex 可以让组件之间的数据共享变得**高效、清晰、且易于维护**。





组件之间的数据共享



黑马程序员
www.itheima.com

传智播客旗下高端IT教育品牌

6. 总结

- 父子关系
 - ① 父 -> 子 属性绑定
 - ② 子 -> 父 事件绑定
 - ③ 父 <-> 子 组件上的 v-model
- 兄弟关系
 - ④ EventBus
- 后代关系
 - ⑤ provide & inject
- 全局数据共享
 - ⑥ vuex



黑马程序员
www.itheima.com

目录 Contents

- ◆ watch 侦听器
- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ vue 3.x 中全局配置 axios
- ◆ 购物车案例

1. 为什么要全局配置 axios

在实际项目开发中，几乎每个组件中都会用到 axios 发起数据请求。此时会遇到如下两个问题：

- ① 每个组件中都需要**导入 axios**（代码臃肿）
- ② 每次发请求都需要填写**完整的请求路径**（不利于后期的维护）



2. 如何全局配置 axios

在 `main.js` 入口文件中，通过 `app.config.globalProperties` 全局挂载 axios，示例代码如下：

```
// 为 axios 配置请求的根路径  
axios.defaults.baseURL = 'http://api.com'  
  
// 将 axios 挂载为 app 的全局自定义属性之后  
// 每个组件可以通过 this 直接访问到全局挂载的自定义属性  
app.config.globalProperties.$http = axios
```

在 `main.js` 入口文件中全局配置 axios

`this.$http.get('/users')`

用户列表组件

`this.$http.get('/profile')`

用户详情组件

`this.$http.post('/news')`

新闻组件

录 Contents

- ◆ watch 侦听器
- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ vue 3.x 中全局配置 axios
- ◆ 购物车案例




购物车案例

1. 案例效果

购物车案例

☒




班俏BANQIAO超火ins潮卫衣女士2020秋季新款韩版宽松慵懒风薄款外套带帽上衣

¥ 108

- 1 +

☒




嘉叶希连帽卫衣女春秋薄款2020新款宽松bf韩版字母印花中长款外套ins潮

¥ 129

- 1 +

☐




思蜜怡2020休闲运动套装女春秋新款时尚大码宽松长袖卫衣两件套

¥ 198

- 1 +

☐




思蜜怡卫衣女加绒加厚2020秋冬装新款韩版宽松上衣连帽中长款外套

¥ 99

- 1 +

☒



露凝早秋季卫衣女春秋装韩版宽松中长款假两件上衣薄款ins盐系外套潮

☐ 全选

合计: ¥ 549.00

结算 (4)





2. 实现步骤

- ① 初始化项目基本结构
- ② 封装 **EsHeader** 组件
- ③ 基于 axios 请求商品列表数据（GET 请求，地址为 <https://www.escook.cn/api/cart>）
- ④ 封装 **EsFooter** 组件
- ⑤ 封装 **EsGoods** 组件
- ⑥ 封装 **EsCounter** 组件





总结

- ① 能够掌握 watch 侦听器的基本使用
 - 定义最基本的 watch 侦听器
 - **immediate**、**deep**、监听对象中单个属性的变化
- ② 能够知道 vue 中常用的生命周期函数
 - 创建阶段、运行阶段、销毁阶段
 - **created**、mounted
- ③ 能够知道如何实现组件之间的数据共享
 - **父子组件**、**兄弟组件**、后代组件
- ④ 能够知道如何在 vue3 的项目中全局配置 axios
 - **main.js** 入口文件中进行配置
 - `app.config.globalProperties.$http = axios`



黑马程序员

www.itheima.com

传智播客旗下高端IT教育品牌

