



黑马程序员™  
www.itheima.com

传智播客旗下  
高端IT教育品牌

# 生命周期 & 数据共享

# 目录 Contents

- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ ref 引用
- ◆ 购物车案例



# 组件的生命周期



黑马程序员  
www.itheima.com

传智播客旗下高端IT教育品牌

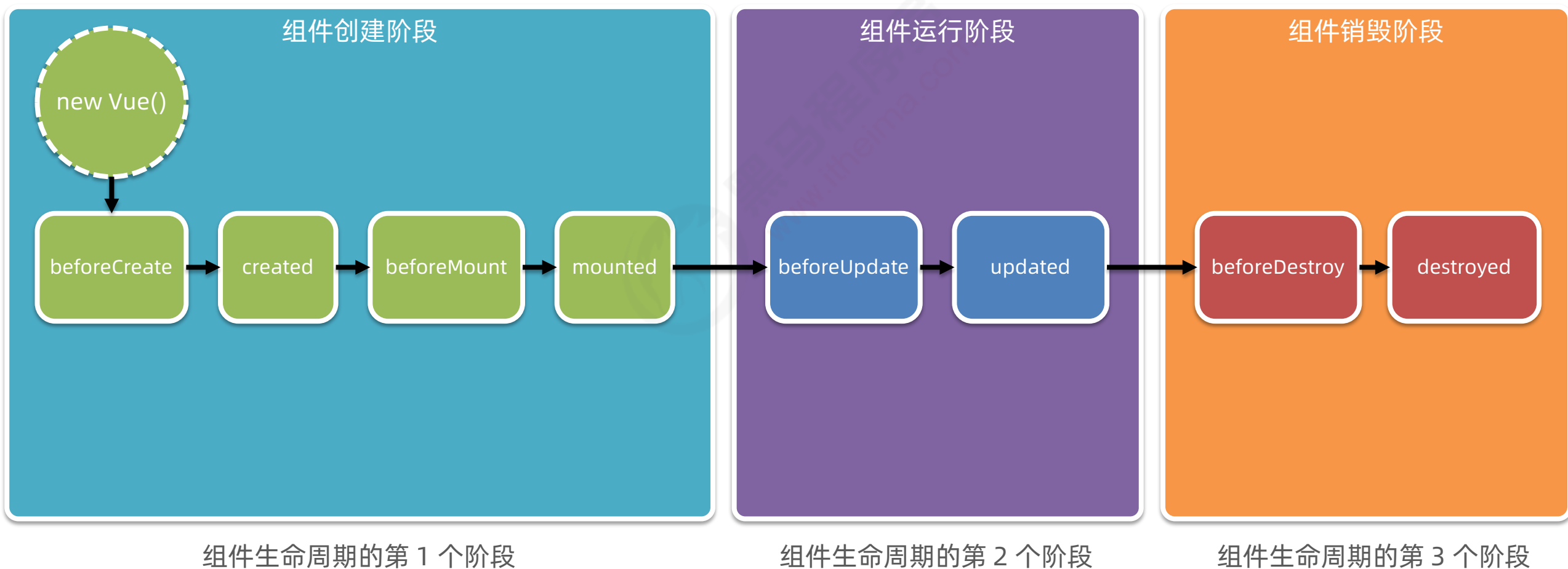
## 1. 生命周期 & 生命周期函数

**生命周期**（Life Cycle）是指一个组件从**创建** -> **运行** -> **销毁**的整个阶段，**强调的是一个时间段**。

**生命周期函数**：是由 vue 框架提供的**内置函数**，会伴随着组件的生命周期，**自动按次序执行**。

注意：**生命周期**强调的是**时间段**，**生命周期函数**强调的是**时间点**。

## 3. 组件生命周期函数的分类





## 4. 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://cn.vuejs.org/v2/guide/instance.html#生命周期图示>



黑马程序员  
www.itheima.com

# 目录 Contents

- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ ref 引用
- ◆ 购物车案例

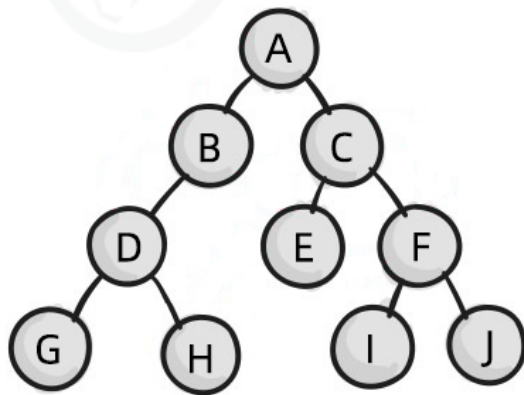


## 1. 组件之间的关系

在项目开发中，组件之间的**最常见的关系**分为如下两种：

- ① 父子关系
- ② 兄弟关系

组件之间的关系





# 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







# 组件之间的数据共享

## 2.1 父组件向子组件共享数据

父组件向子组件共享数据需要使用自定义属性。示例代码如下：

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

```
1 <template>
2   <div>
3     <h5>Son 组件</h5>
4     <p>父组件传递过来的 msg 值是: {{ msg }}</p>
5     <p>父组件传递过来的 user 值是: {{ user }}</p>
6   </div>
7 </template>
8
9 props: ['msg', 'user']
```



# 组件之间的数据共享

## 2.2 子组件向父组件共享数据

子组件向父组件共享数据使用自定义事件。示例代码如下：

子组件

```
1 export default {
2   data() {
3     return { count: 0 }
4   },
5   methods: {
6     add() {
7       this.count += 1
8       // 修改数据时, 通过 $emit() 触发自定义事件
9       this.$emit('numchange', this.count)
10    }
11  }
12 }
```

父组件

```
1 <Son @numchange="getNewCount"></Son>
2
3 export default {
4   data() {
5     return { countFromSon: 0 }
6   },
7   methods: {
8     getNewCount(val) {
9       this.countFromSon = val
10    }
11  }
12 }
```



# 组件之间的数据共享

## 3. 兄弟组件之间的数据共享

在 **vue2.x** 中，兄弟组件之间数据共享的方案是 **EventBus**。

```
import bus from './eventBus.js'

export default {
  data() {
    return {
      msg: 'hello vue.js'
    }
  },
  methods: {
    sendMsg() {
      bus.$emit('share', this.msg)
    }
  }
}
```

兄弟组件 A（数据发送方）

```
import Vue from 'vue'

// 向外共享 Vue 的实例对象
export default new Vue()
```

eventBus.js

```
import bus from './eventBus.js'

export default {
  data() {
    return {
      msgFromLeft: ''
    }
  },
  created() {
    bus.$on('share', val => {
      this.msgFromLeft = val
    })
  }
}
```

兄弟组件 C（数据接收方）



# 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

## EventBus 的使用步骤

- ① 创建 `eventBus.js` 模块，并向外共享一个 `Vue` 的实例对象
- ② 在数据发送方，调用 `bus.$emit('事件名称', 要发送的数据)` 方法触发自定义事件
- ③ 在数据接收方，调用 `bus.$on('事件名称', 事件处理函数)` 方法注册一个自定义事件

# 目录 Contents

- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ ref 引用
- ◆ 购物车案例

## 1. 什么是 ref 引用

ref 用来辅助开发者在**不依赖于 jQuery 的情况下**，获取 DOM 元素或组件的引用。

每个 vue 的组件实例上，都包含一个 **\$refs 对象**，里面存储着对应的 DOM 元素或组件的引用。默认情况下，组件的 **\$refs** 指向一个空对象。

```
1 <template>
2   <div>
3     <h3>MyRef 组件</h3>
4     <button @click="getRef">获取 $refs 引用</button>
5   </div>
6 </template>
7
8 export default {
9   methods: {
10     getRef() { console.log(this) } // this 是当前组件的实例对象, this.$refs 默认指向空对象
11   }
12 }
```

## 2. 使用 ref 引用 DOM 元素

如果想要使用 ref 引用页面上的 DOM 元素，则可以按照如下的方式进行操作：

```
1 <!-- 使用 ref 属性，为对应的 DOM 添加引用名称 -->
2 <h3 ref="myh3">MyRef 组件</h3>
3 <button @click="getRef">获取 $refs 引用</button>
4
5 methods: {
6   getRef() {
7     // 通过 this.$refs.引用的名称 可以获取到 DOM 元素的引用
8     console.log(this.$refs.myh3)
9     // 操作 DOM 元素，把文本颜色改为红色
10    this.$refs.myh3.style.color = 'red'
11  },
12 }
```

### 3. 使用 ref 引用组件实例

如果想要使用 ref 引用页面上的组件实例，则可以按照如下的方式进行操作：

```
1 <!-- 使用 ref 属性，为对应的“组件”添加引用名称 -->
2 <my-counter ref="counterRef"></my-counter>
3 <button @click="getRef">获取 $refs 引用</button>
4
5 methods: {
6   getRef() {
7     // 通过 this.$refs.引用的名称 可以引用组件的实例
8     console.log(this.$refs.counterRef)
9     // 引用到组件的实例之后，就可以调用组件上的 methods 方法
10    this.$refs.counterRef.add()
11  },
12 }
```



## 4. 控制文本框和按钮的按需切换

通过布尔值 `inputVisible` 来控制组件中的文本框与按钮的按需切换。示例代码如下：

```
1 <template>
2   <input type="text" v-if="inputVisible">
3   <button v-else @click="showInput">展示input输入框</button>
4 </template>
```

```
1 <script>
2 export default {
3   data() {
4     return {
5       // 控制文本框和按钮的按需切换
6       inputVisible: false,
7     }
8   },
9   methods: {
10    showInput() { // 切换布尔值，显示文本框
11      this.inputVisible = true
12    },
13  },
14 }
15 </script>
```

## 5. 让文本框自动获得焦点

当文本框展示出来之后，如果希望它立即获得焦点，则可以为其添加 ref 引用，并调用原生 DOM 对象的 .focus() 方法即可。示例代码如下：

```
1 <input type="text" v-if="inputVisible" ref="ipt">
2 <button v-else @click="showInput">展示input输入框</button>
3
4 methods: {
5   showInput() {
6     this.inputVisible = true
7     // 获取文本框的 DOM 引用，并调用 .focus() 使其自动获得焦点
8     this.$refs.ipt.focus()
9   },
10 }
```

## 6. this.\$nextTick(cb) 方法

组件的 `$nextTick(cb)` 方法，会把 `cb` 回调推迟到下一个 DOM 更新周期之后执行。通俗的理解是：等组件的 DOM 更新完成之后，再执行 `cb` 回调函数。从而能保证 `cb` 回调函数可以操作到最新的 DOM 元素。

```
1 <input type="text" v-if="inputVisible" ref="ipt">
2 <button v-else @click="showInput">展示input输入框</button>
3
4 methods: {
5   showInput() {
6     this.inputVisible = true
7     // 把对 input 文本框的操作，推迟到下次 DOM 更新之后。否则页面上根本不存在文本框元素
8     this.$nextTick(() => {
9       this.$refs.ipt.focus()
10    })
11  },
12 }
```

# 目录 Contents

- ◆ 组件的生命周期
- ◆ 组件之间的数据共享
- ◆ ref 引用
- ◆ 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

+

☒



嘉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

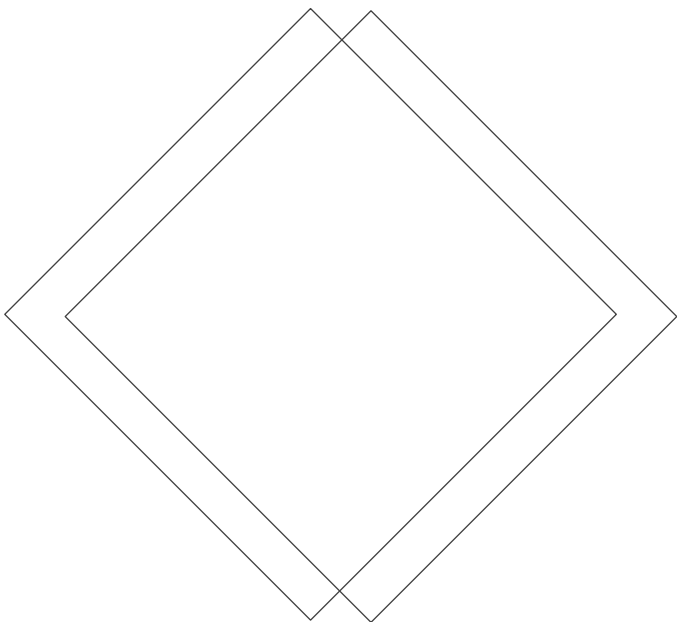
合计: ¥ 549.00

结算 (4)



## 2. 实现步骤

- ① 初始化项目基本结构
- ② 封装 **MyHeader** 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 **MyFooter** 组件
- ⑤ 封装 **MyGoods** 组件
- ⑥ 封装 **MyCounter** 组件



## ① 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**

## ② 能够知道如何实现组件之间的数据共享

- 父 -> 子 (**自定义属性**)
- 子 -> 父 (**自定义事件**)
- 兄弟组件 (**EventBus**)

## ③ 能够知道如何使用 ref 引用 DOM 元素或组件

- 给元素或组件添加 **ref="xxx"** 的引用名称
- 通过 **this.\$refs.xxx** 获取元素或组件的实例
- **\$nextTick()** 函数的执行时机



传智播客旗下高端IT教育品牌