# Vue的基本使用2

## vue的组件

### 组件的使用

#### 组件之间的父子关系



组件在被封装好之后，彼此之间时相互独立的，不存在父子关系



在使用组件的时候，根据彼此的嵌套关系，形成了父子关系，兄弟关系

#### 使用组件的三个步骤

例如这里将app.vue作为父组件

left.vue和right.vue作为子组件

1.



现在app.vue组件中使用import语法导入需要的组件（例如这里的left）

2.



使用components节点注册组件

3.



以标签的形式使用刚才注册的组件

可以理解为创建一个自定义的标签

注意：通过components注册的是私有子组件

例如：

在组件A的components节点下，注册了组件F

则组件F只能用在组件A中；不能用在组件C中

#### 使用Vue.component全局注册组件

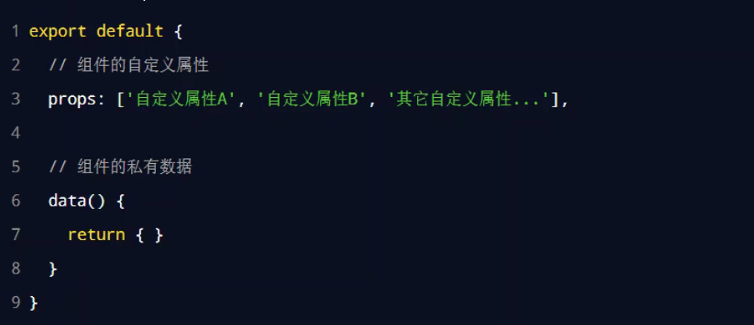
在vue项目的main.js入口文件中，通过Vue.component()方法，可以注册全局组件。示例代码如下：



#### 组件的props自定义属性

props是组件的自定义属性，在封装通用组件的时候，合理地使用props可以极大提高组件的复用性！（例如不同的父组件调用该组件时，对同一个数据要赋予不同的值）

语法格式如下：



**基础使用：**

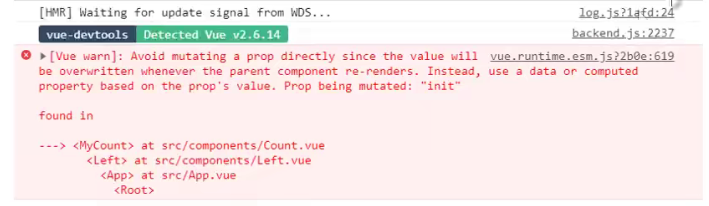
在调用的时候之间赋值：



这里加v-bind是因为想让init的值等于9，调用v-bind使后面的赋值为js语句

**注意**：

props是只读的，在组件封装的自定义属性是只读的，程序员不能直接修改props的值。否则会直接报错！



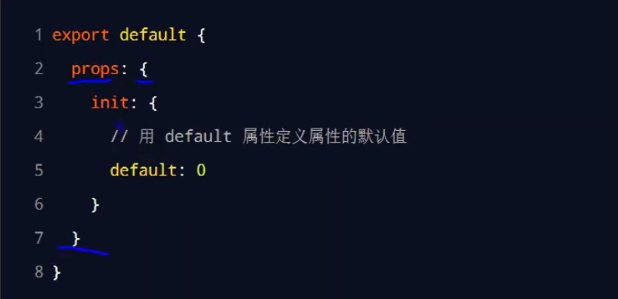
若想要修改props的值，可以把props的值转存到data中，因为data中的数据是可读可写的！



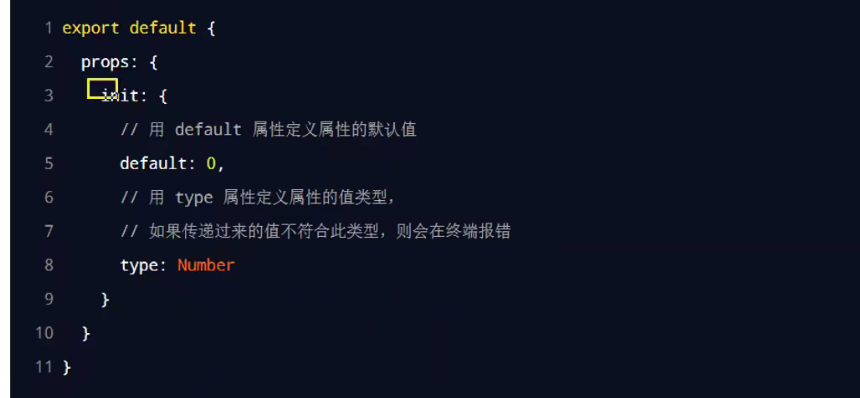
**props中的自定义规定**

在声明自定义属性时，也可以通过对象的方式对自定义的属性进行更细化的规定

例如可以用 default 来规定自定义属性的默认值：



使用type来规定数据类型：



使用required来规定数据是否为必填：

#### 组件之间的样式冲突

默认情况下，写在 .vue 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。

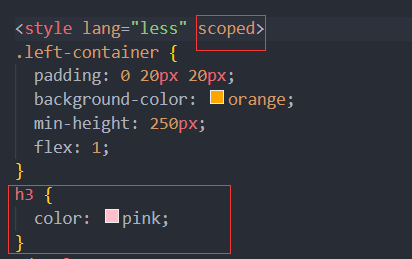
导致组件之间样式冲突的根本原因是：

1. 单页面应用程序中，所有组件的DOM结构，都是基于唯一的index.html页面进行呈现的
2. 每个组件中的样式，都会影响整个index.html页面中的DOM元素

如何解决：

1.同级组件之间：在style标签中加入scoped属性即可

scoped属性的作用是在当前组件的标签中自动生成一个data-v的自定义属性，在定义css样式时也会通过子选择器自动添加上添加的自定义属性，从而防止同级组件之间的样式冲突



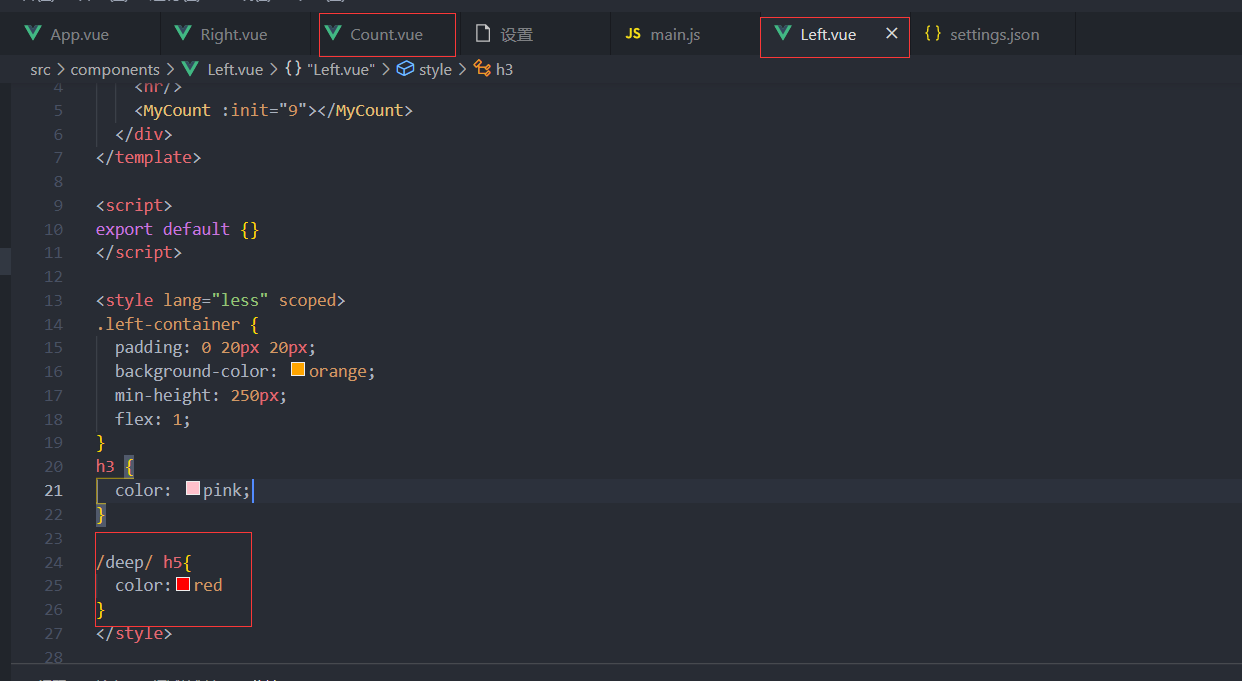


2. /deep/ 样式穿透

当父组件中想修改子组件中的DOM元素样式时可以使用 /deep/ 进行样式穿透进行修改

示例：

在父组件left修改count组件的元素样式：



浏览器中：

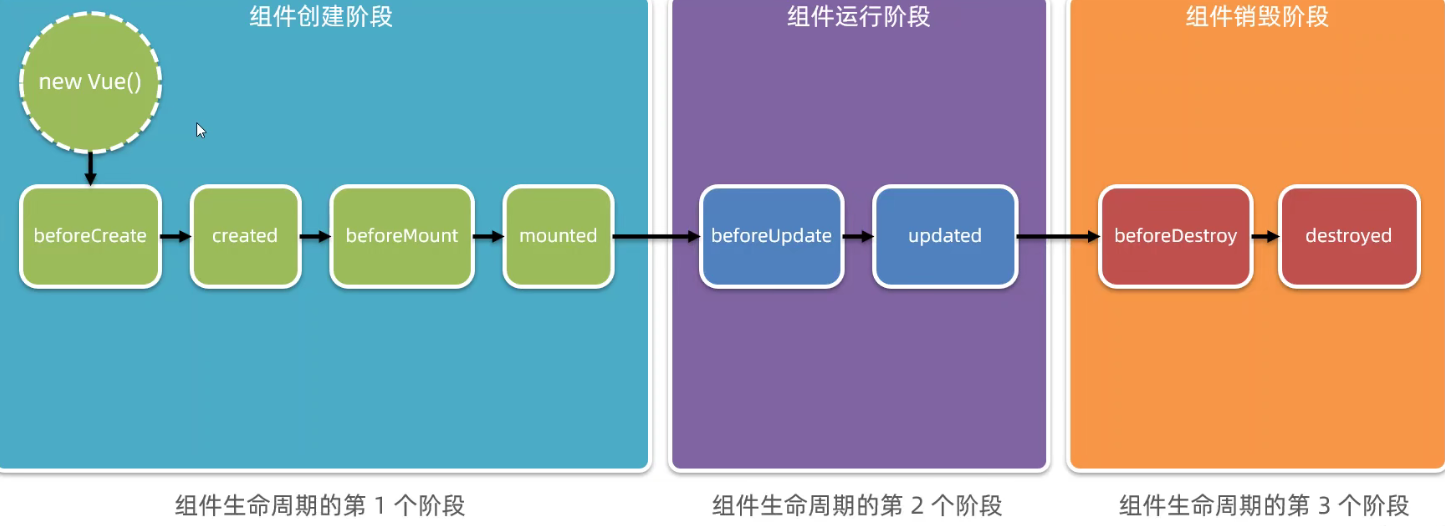


## 组件的生命周期

生命周期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，会伴随组件的生命周期，自动按次序执行

### 组件生命周期的分类：



created：创建，在模板渲染成html前调用，即通常初始化某些属性值，然后再渲染成视图。

mount：在模板渲染成html后调用，通常是初始化页面完成后，再对html的dom节点进行一些需要的操作。

update：更新

destroy：销毁

### 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#](https://cn.vuejs.org/v2/guide/instance.html)生命周期图示



### 生命周期各阶段生命周期函数的使用

#### beforecreate和created生命周期函数的使用：

1. beforecreate生命周期函数：

在当前阶段组件的props/data/methods等节点内容尚未被创建，都处于不可用状态，基本无用

1. created生命周期函数：

在当前阶段组件的props/data/methods节点内容已经初始化完毕，都处于可用状态，但是组件的模板结构尚未生成。

这个阶段是最常用的，因为节点初始化的完成，可以在当前阶段进行ajax请求向服务端发送请求，并将请求来的数据转存到data中

#### beforemount和mounted生命周期函数的使用：

1. beforemount生命周期函数：

基本无用

1. mounted生命周期函数：

开始可以操作DOM，也是最早能操作DOM元素的阶段

#### 组件运行阶段的生命周期函数beforeUpdate和updated

1. beforeUpdate生命周期函数：

将要根据变化过后、最新的数据，重新渲染组件的模板结构，所以至少在数据发送变化时触发，data/props/methods等节点内的数据会更新，但是页面内容不会变化

1. updated生命周期函数：

数据更新且页面内容也更新

当数据发生变化后，为了能够操作到最新的DOM结构，必须把代码写到updata生命周期函数中

#### beforeDestroy和destroyed生命周期函数

例如使用v-if进行组件的销毁

1. beforeDestroy生命周期函数：

将要销毁此组件，此时尚未销毁，组件还处于正常工作的状态

1. destroyed生命周期函数：

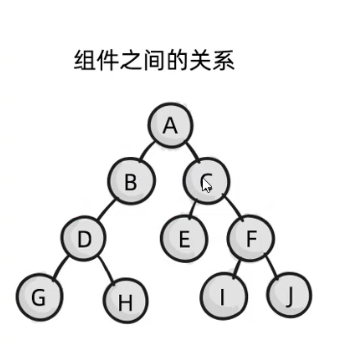
组件已经被销毁，此组件在浏览器中对应的DOM结构已经被完全移除

## 组件之间共享数据

### 组件之间的关系

在项目开发中，组件之间的最常见关系分为如下两种：

1. 父子关系
2. 兄弟关系



#### 1.1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：



注意：不要修改props定义属性的值，需要修改时在data内定义一个属性用处储存和修改

#### 1.2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：



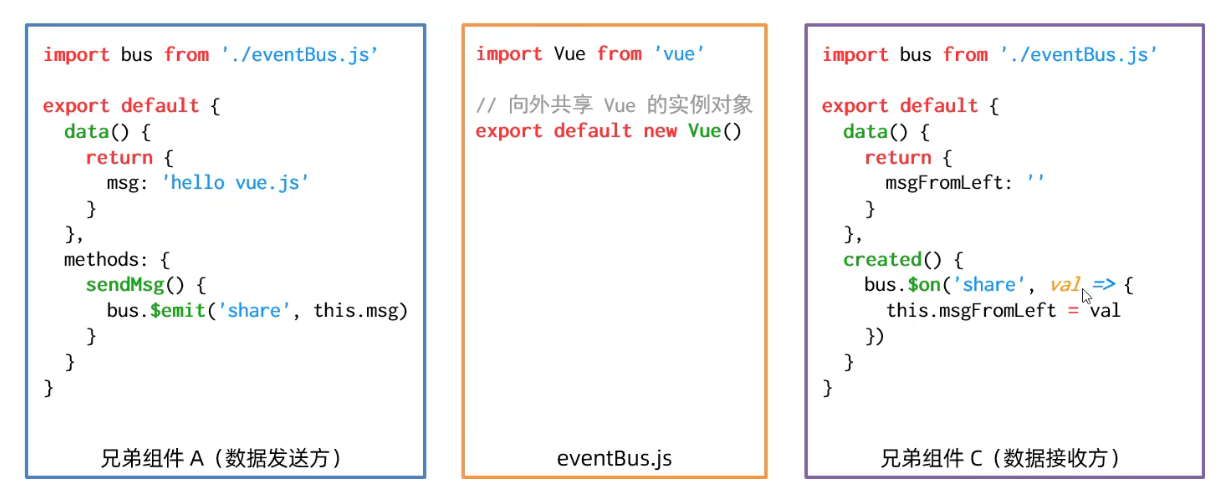
$emit()方法将子组件的数据发送给父组件，第一个参数是自定义事件，第二个是需要传递的数据

在父组件中要调用子组件自定义的事件，并设置一个事件处理函数用来接收数据

上面的示例中当子组件的按钮触发时就会将子组件的数据发送给父组件

#### 1.3兄弟组件之间共享数据

在vue2.x中，兄弟组件之间的数据共享的方案是EventBus



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. 在数据方，导入模块后（假设导入名为bus），调用bus.$emit(‘事件名称’,要发送的数据)方法触发自定义事件
3. 在数据接收方，导入模块后(同样假设导入名为bus)，调用bus.$on(‘事件名称’,事件处理函数)的方法注册一个自定义事件