# vue day5

## 动态组件

1. 什么是动态组件：

动态组件指的是动态切换组件的显示与隐藏

1. 如何实现动态组件的渲染

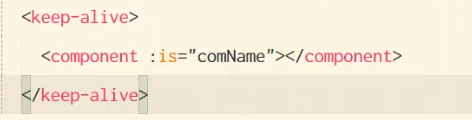
vue提供了一个内置的<component>组件，专门用来实现动态组件的渲染。示例代码如下：



通过修改comName的值可以进行组件切换，每一次切换都会销毁前一次的组件

1. 使用keep-alive保持状态

使用keep-alive可以防止组件被隐藏的时候被销毁，示例：



1. keep-alive对应的生命周期函数

当组件被缓存的时候，会自动触发组件的deactivated生命周期函数

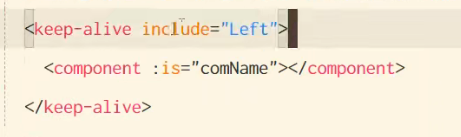
当组件被激活的时候，会自动触发组件的activated生命周期函数

当组件第一次创建的时候会同时触发created和activated两个生命周期函数

但是当再次激活时，只会触发activated生命周期函数

1. keep-alive的include属性

include属性用来指定：只有名称匹配的组件会被缓存。多个组件之间使用英文的逗号隔开，示例：

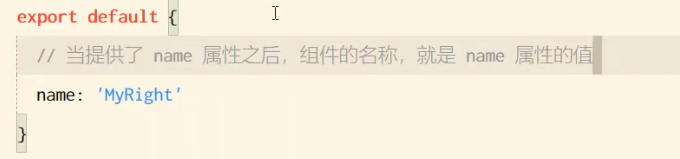


也有exclude属性：名称匹配的组件不被缓存，多个组件间用逗号分隔

注：exclude和include只能使用一个，不能同时存在

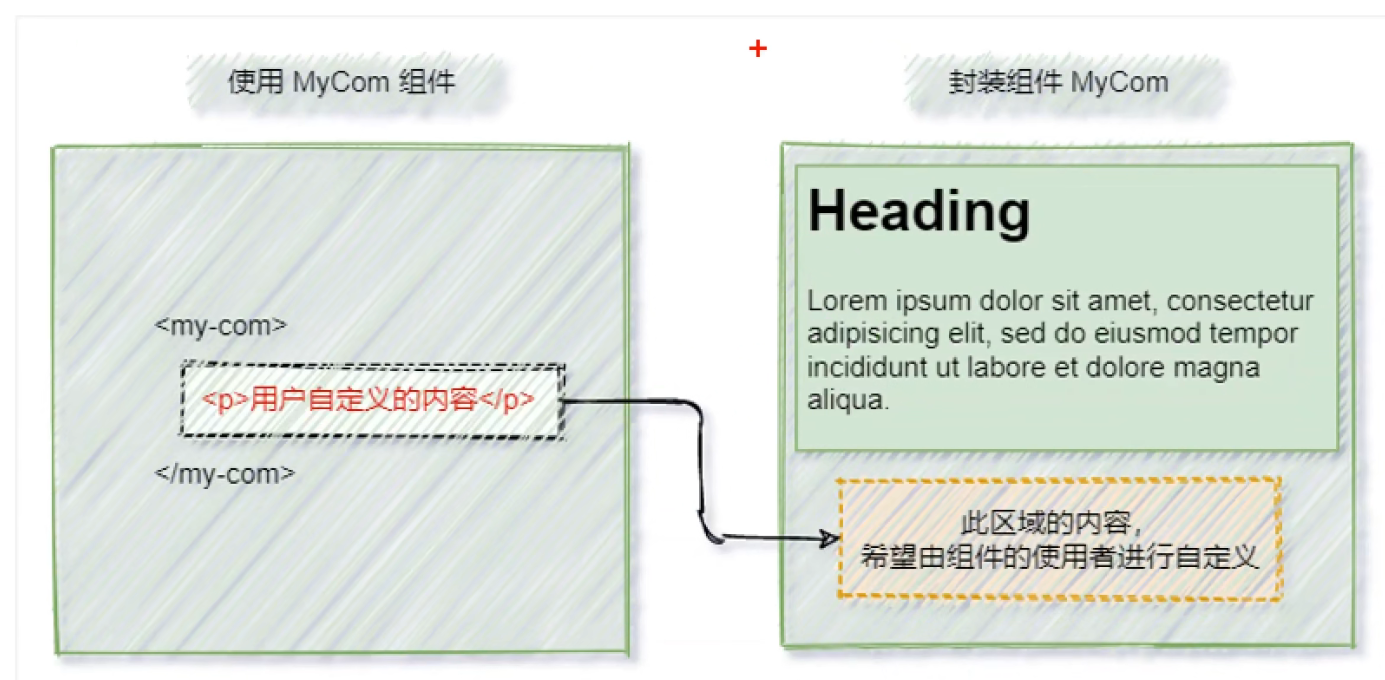
1. 组件声明时name属性

当组件声明时没有使用name属性，则默认组件名为注册名，若有定义name属性则按照name属性的属性值规定组件名，例：



## 插槽

1. 插槽（slot）时vue为组件的封装者提供的能力。允许开发者再封装组件时，把不确定的、希望用户指定的部分定义为插槽



演示：

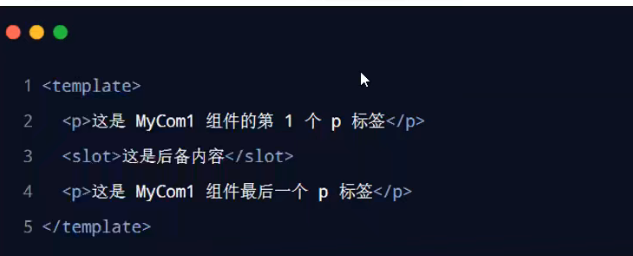


v-slot可以进行简写，简写的方式为：#

<slot>标签内可以写东西，作为该插槽的默认值，当有组件用到该插槽时默认值会被覆盖

1. 后备内容

封装组件时，可以为预留的<slot>插槽提供后备内容（默认内容）。如果组件的使用者没有为该插槽提供任何内容，则后备内容生效。示例：



1. 具名插槽

即每个插槽有对应的插槽名，对应不同的内容

在插槽中还有作用域插槽，即可以在插槽内自定义属性，在使用插槽的时候可以进行接收与使用，例：

定义插槽



使用插槽



scope为形参，用来接收msg的值，这个形参可以自定义名称

## 自定义指令

1. 什么是自定义指令

vue官方提供了v-text、v-for、v-model、v-if等常用的指令指令外。vue还允许开发者自定义指令

1. 自定义指令的分类

vue中又两种自定义指令：

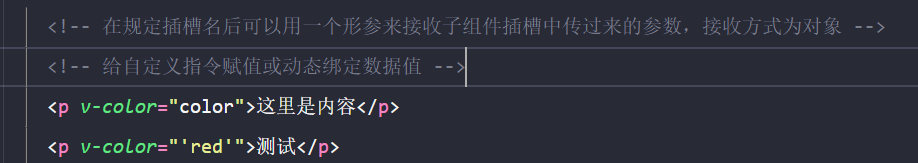
1. 私有自定义指令
2. 全局自定义指令
   1. 私有自定义指令

在每个vue组件中，可以在directives节点下声明私有自定义指令，示例如下：



* + 1. 使用binding.value获取指令绑定的值

当绑定了自定义指令时，可以给指令赋值，在bind函数中用binding形参接收值，例：





* + 1. updata函数

bind函数只调用1次，当指令第一次绑定到元素时调用，当DOM更新时bind函数不会触发。updata函数会在每次DOM更新时调用。示例代码：

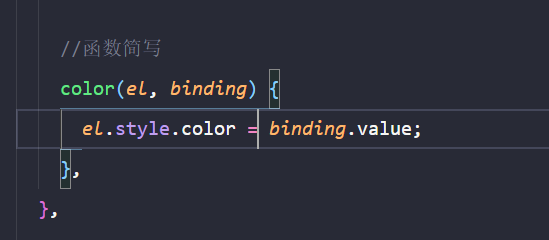


* + 1. 函数简写

如果bind和updata函数中的逻辑完全相同，则对象格式的自定义指令可以简写成函数形式，例：



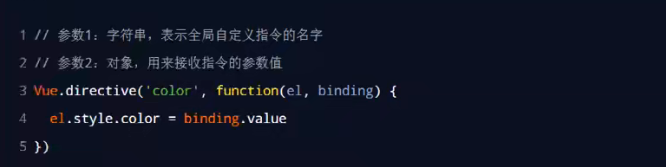
bind和updata逻辑完全相同



简写

* 1. 全局自定义指令

全局共享的自定义指令需要通过“vue.directive()”进行声明，示例如下：



## ESLint