# Vue 组件

## 什么是组件

组件是Vue.js最强大的功能之一。组件可以扩展HTML元素，封装可重用的代码。在较高层面上，组件是自定义的元素，Vue.js的编译器为它添加特殊功能。在有些情况下，组件也可以是原生HTML元素的形式，以js特性扩展。

## 组件的介绍

有多种方式可以在vue中定义模板组件。这里给大家简单介绍几种定义组件的方式，比如字符串（String）、模板字符串（Template literal）、X-Templates、单文件组件等，我们项目中用的比较多的一种定义方式就是单文件组件的方式，我们这里主要是讲解单文件组件方式定义组件，至于其余的组件定义方式，我们并不做重点的介绍，只是作为一个了解内容而已。下面我们就依次来讲解一下各种组件的定义方式及其特点：

## 字符串方式

默认情况下，在JS文件里模板会被定义为一个字符串。但是我觉得大家都会同意这种写法很难看懂，它除了有广泛的浏览器支持之外，并没有什么优势。下面我们看一下字符串方式定义组件的代码：

Vue.component('my-checkbox', {

  template:'<div class="checkbox-wrapper" @click="check"><div :class="{ checkbox: true, checked: checked }"></div><div class="title">{{ title }}</div></div>',

  data() {

return {

checked: false,

title: 'Check me'

}

  },

  methods: {

    check() { this.checked = !this.checked; }

  }

});

## 模板字符串方式

通过ES6的模板字符串（反引号）语法，你在定义模板时可以直接换行，这是通过常规的JavaScript字符串没法做到的。这种写法更容易阅读，并且这种模板字符串语法得到了许多新版本浏览器的支持。当然，为了安全起见，你仍然应该把它转译为ES5的语法形式。

缺点还是比较明显的：这种方式并不完美，我发现大多数的IDE在语法高亮上做的差强人意，并且在缩进和换行等的格式方面，仍然很痛苦。

Vue.component('my-checkbox', {

template: `<div class="checkbox-wrapper" @click="check">

<div :class="{ checkbox: true, checked: checked }"></div>

<div class="title">{{ title }}</div>

</div>`,

data() {

return { checked: false, title: 'Check me' }

},

methods: {

check() { this.checked = !this.checked; }

}

});

## X-Templates 方式

使用这种方法，你需要在index.html文件里的script标签中定义你的模板。script标签需要添加text/x-template类型作为标记，并且在定义组件时，通过id来引用。

说实话这种定义方式使用起来还是比较舒服的，它允许你使用真正的HTML标记来书写你的HTML文件，但是不足之处在于，这种方式会把模板和组件其它部分的定义分开。

Vue.component('my-checkbox', {

template: '#checkbox-template',

data() {

return { checked: false, title: 'Check me' }

},

methods: {

check() { this.checked = !this.checked; }

}

});

<script type="text/x-template" id="checkbox-template">

<div class="checkbox-wrapper" @click="check">

<div :class="{ checkbox: true, checked: checked }"></div>

<div class="title">{{ title }}</div>

</div>

</script>

基于以上组件的介绍，这些方式在很多中小规模的项目中运作的很好，在这些项目里 JavaScript 只被用来加强特定的视图。但当在更复杂的项目中，或者你的前端完全由 JavaScript 驱动的时候，下面这些缺点将变得非常明显：

1、全局定义 (Global definitions) 强制要求每个 component 中的命名不得重复

2、字符串模板 (String templates) 缺乏语法高亮，在 HTML 有多行的时候，需要用到丑陋的 \

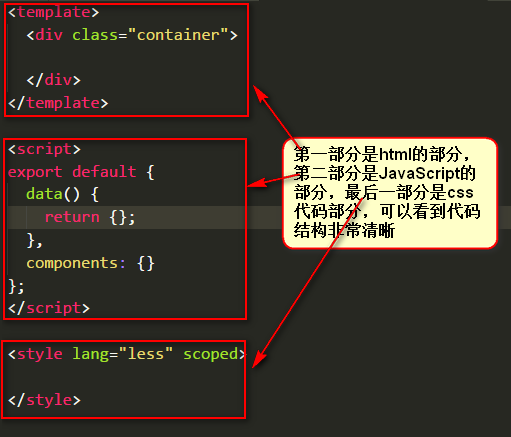
3、不支持 CSS (No CSS support) 意味着当 HTML 和 JavaScript 组件化时，CSS 明显被遗漏

4、没有构建步骤 (No build step) 限制只能使用 HTML 和 ES5 JavaScript, 而不能使用预处理器，如Babel

## 单文件组件方式

为了解决以上问题，单文件组件应运而生，文件扩展名为 .vue 的 single-file components(单文件组件) 为以上所有问题提供了解决方法，并且还可以使用 webpack 或 Browserify 等构建工具。我们项目中就比较多的使用了单文件组件，构建工具我们使用的是webpack工具

下面我们就来看一个简单的单文件组建的代码结构：

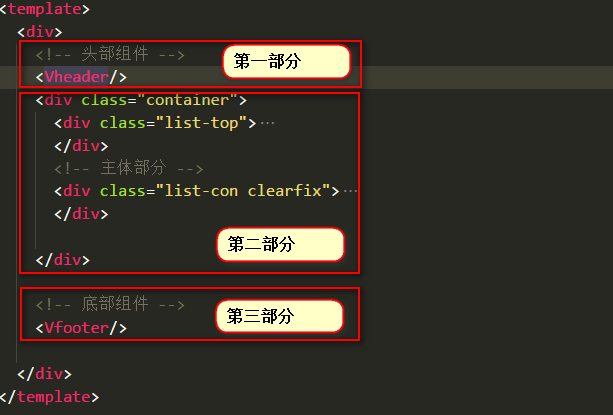


通过上面一张图片我们现在我们获得：

1. 完整语法高亮
2. CommonJS 模块
3. 组件作用域的 CSS

正如我们说过的，我们可以使用预处理器来构建简洁和功能更丰富的组件，比如 Babel (with ES2015 modules)，和less等。

介绍了这么多，下面我们就就结合我们项目中的单文件组件给大家做介绍，看一下我们的实际项目中是如何来操作的,我们还是以列表页为例：



这是我们商品列表页面的的template的区域，也就是我们页面的html结构部分，下面我们就分块给大家介绍一下这块的结构部分：

第一部分：首先要给你大家介绍组件的比较重要的一个作用就是复用性，一个项目做下来，肯定会有复用的东西，最简单的比如说项目当中页面的头部和尾部，这一部分的内容一般都是复用性不交高的部分，所以这块的内容一般都是以单个组件的形式出现的，以方便可以一次完成，随处可以调用，这一部分的作用就是引入网站的头部组件

第二部分：这一块内容每个页面的复用度不是很大，一般都是自己所特有的，所以就是自己自行使用标签布局完成的，这部分的代码操作方式和我们平时开发一样，这部分就是结构操作，至于具体的样式控制，会有单独的地方去完成，也比较符合W3C的结构和样式分离的规范

第三部分：这块的操作和第一部分操作基本一样的，底部基本也是一样的，所以底部的部分我们也是单独写成了一个单文件组件来方便调用；

下面我们看一下JavaScript部分的代码：



下面我们来具体的分析一下JavaScript部分：

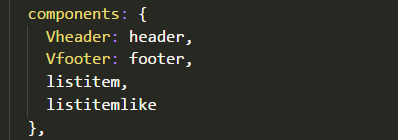
第一个红框内的代码表示我们当前的商品列表页面需要用到的外部的单文件组件，比如这里引用的头部，底部组件，还有商品的列表组件，这一块的商品列表也是重用性比较高的的部分，我们从后台的数据库拿到商品的对象信息，然后循环遍历出商品列表，这也属于复用度比较高的部分了，所以也是抽离为单文件组件了；

第二个红框内的代码为具体的JavaScript代码部分，这一块做一个详细的介绍

首先 export default 是ES6的语法，他的主要作用是导出一个对象，这个大的对象里面包括了data、组件、生命周期函数定义使用、事件方法的使用等等下面我们一次来说下

首先是data，data这块是以一个函数的形式来完成的，里面return的是一系列的变量，这些变量的作用和我们传统开发当中变量的作用完全一致，就是一些接收传递参数的作用，在使用变量的时候，一定要先在这里做定义才可以使用，不然的话，会提示一些警告，变量在使用前先声明再使用这是一种规范的做法，当然在vue的开发中我们依然要这样去完成我们的变量的操作；

然后是components，里面是我们页面所需要的组件，组件首先是在开始的部分要 先import导入到我们当前的页面，然后在components里面做登记，才能最后在页面当中使用了，看到下面途中的截图可以知道，登记组件可以是json格式的形式引入也可以直接引入名字，这个地方有个需要注意的点，比如我们组件命名为header，如果不用json格式的方式引入给他重命名的话，我们直接在template中使用<header>引入这个单文件组件的话，就会和HTML5当中的标签冲突，所以我们组件命名的时候语义化 是一方面但是在使用的时候一定不能和标签名字冲突，Vue当中是有明确规定，所以这块大家需要有一个重点的掌握，尽量避开这一点；



接下来是mounted方法，这是vue中的生命周期函数，对于生命周期这块的介绍，会有专门的文档去介绍，这里只是针对mounted方法做介绍，mounted是代表生命周期当中组件刚刚加载之后的一个钩子函数，比如我们商品列表也刚刚加载完之后，我们希望去后台获得数据渲染商品列表，那么这时候就用到了mounted函数，把这一系列的操作都会放到这里做完成，关于更多的生命周期的介绍这里就不过多介绍了，更为详细的介绍可以取查看vue生命周期的文档；

最后一部分是methods，这一块存放的是具体的事件操作，不如我们点击某个按钮触发事件的时候，我们这个函数就要写到methods里面，这里是统一存放事件操作函数的地方，这里需要给大家申明一下，如果页面逻辑比较复杂的话，事件函数会比较多，大家在这里一定要写好注释，注释清楚每一个事件函数的具体作用，以便后期查找的时候有个提醒，可以更好的去帮组自己回忆；

## 总结

关于vue组件的部分这里就简单的介绍了一下，组件这块是vue框架中的重点，这里我们把主要的部分放到了当文件组件的讲解上，因为单文件组件的使用确实比较舒服，能够避免了很多鸡肋的东西，在我们项目中应用是非常广泛的，希望大家一点要着重掌握单文件组件的使用。