**Vue简单入门**

**Vue概述与快速入门**

**VueJS介绍**

Vue.js是一个构建数据驱动的web界面的渐进式框架，Vue.js的目标是通过尽可能简单的API实现响应的数据绑定和组合的视图组件。它不仅易于上手，还便于与第三方库或既有项目整合。Vue是目前前端最后的一个前端框架，加分项[高薪必备]。

官网：<https://cn.vuejs.org/>

提高开发效率的发展历程:

原生JS-Jquery之类的类库--前端模板引擎[整个页面渲染数据]--Angular.js/**vue.js**

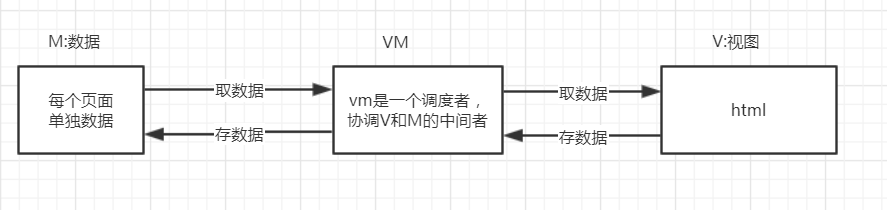
**MVVM模式**

MVVM是Model-View-ViewModel的简写。它本质上就是MVC的改进版。MVVM就是将其中的View状态和行为抽象化，让我们将视图UI和业务逻辑分开。

MVVM模式和MVC模式一样，主要目的是分离视图(View)和模型(Model)。

Vue.js是一个提供了MVVM风格的双向数据绑定的Javascript库，**专注于View层**。**它的核心是MVVM中的VM,也就是ViewModel。ViewModel负责连接View和Model，保证视图和数据的一致性**，这种轻量级的架构让前端开发更加高效、便携。

Vue的核心理念是：让用户不再操作DOM元素，解放用户双手、让开发人员更多的去关注业务逻辑本身。



注意：**MVVM的双向数据绑定是由VM决定的**。

**Vue入门案例**

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script type="text/javascript" src="static/vue/vuejs-2.5.16.js"></script>

</head>

<body>

<div id="app">

{{message}} <!-- Vue的插值表达式，把data中定义的数据显示到此处 -->

</div>

<script>

//导入了Vue.js之后，内存中就有了Vue的构造函数，

var vm= new Vue({

el:"#app", //由Vue接管id为app的区域

data:{

message:"Hello Vue" //注意：此处不要加分号

}

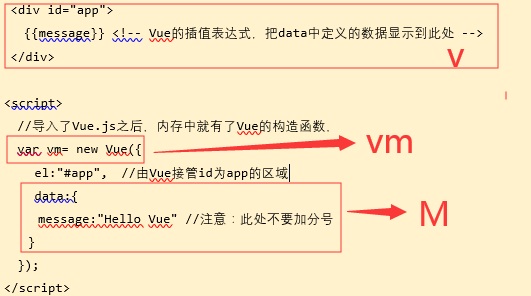
});

</script>

</body>

写好之后就可以在浏览器中打开访问了。

其中：MVVM指：



**插值表达式**

数据绑定最常见的形式就是使用”Mustache”语法(双大括号)的文本插值，Mustache标签讲话呗替代为对应数据对象上属性的值。无论何时，绑定的数据对象上属性发生了改变，插值处的内容都会更新。

Vue.js都提供了完全的JavaScript表达式支持。

{{ number + 1}}

{{ ok ? ‘yes’:’No’}}

这些表达式会在所属Vue实例的数据作用域下作为JavaScript被解析。有个限制就是，每个绑定都只能包含**单个表达式**，所以下面的例子都**不会**生效。

<!-- 这是语句，不是表达式-->

{{var a = 1}}

<!-- 流控制也不会生效，请使用三元表达式 -->

{{if(ok){ return message }}}

**VueJS常用系统指令**

指令都是以v-开头的。

**v-cloack**

v-cloack:解决出现插件符号的问题,当网速过慢的情况下，比如：slow 3G模式

<body>

<!-- VUE控制的这个区域就是我们所说的 V -->

<div id="app01">aaa

<h1>{{msg}}</h1>

</div>

<script type="text/javascript" src="static/js/vuejs-2.5.16.js"></script>

<script>

var vm= new Vue({

el:'#app01',

data:{

msg: '欢迎来到vue学习课堂'

}

});

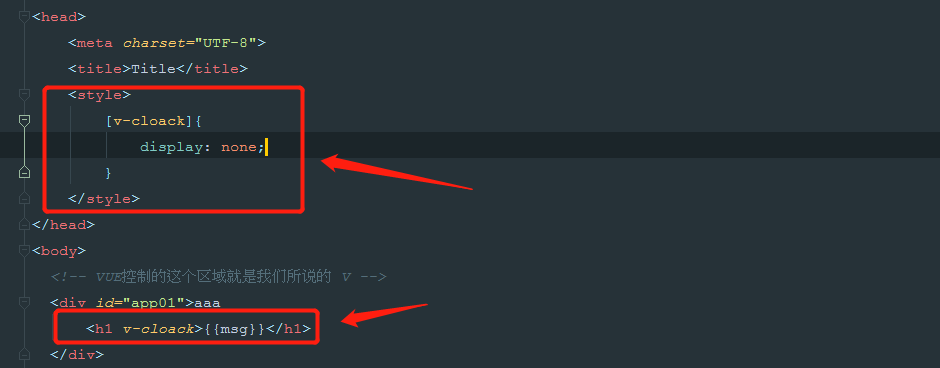
</script>

页面上请求的数据没回来的时候，页面会

先显示：{{ }} 插值表达式

再显示：请求回来的数据。

那怎么解决这种问题呢？可以在<h1> 标签使用v-cloack属性，然后使得凡是加了v-cloack属性的元素都隐藏起来。如下所示：



所以：v-cloack可以解决插值表达式的闪烁问题。

**v-text**

使用：

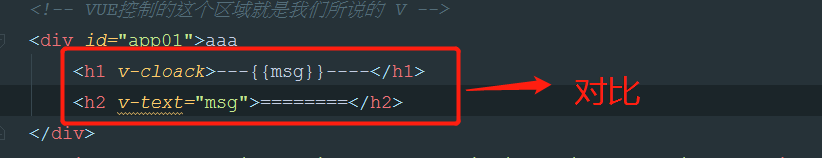


v-text属性和插值表达式的作用类似，都是在标签内部显示某数据信息。

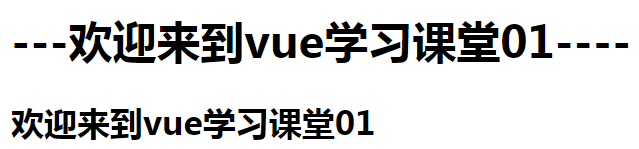
不同：

v-text没有闪烁问题，但是v-text也会覆盖标签内部原本的内容。

插值表达式只会替换自己的占位符，不会把整个标签内部的内容都清空。



效果：



**v-html**

data的数据以html的样式解析，如下所示：

*<!-- VUE控制的这个区域就是我们所说的 V -->* <div *id*="app01">
<div>{{msg}}</div>
<d*iv v-te*xt="msg"></div>
<*div v-h*tml="msg"></div>
</div>
<scr*ipt>*
*var* vm= new Vue({
el:'#app01',
data:{
msg: '<h1>beautiful girl<h1>'
}
});
</script>

效果：



**v-bind**

v-bind命令是vue中提供的一个用于**绑定属性**的指令，**v-bind:**可以简写为**:[冒号]** ,演示如下：



如果想要使用data中的12，必须使用v-bind命令将data中的变量绑定到input的value属性上。



当然v-bind会把后面的age当做**变量**去data中找对应的值，而且我们还可以给变量加上一个值,也即**v-bind指令中可以写合法的js表达式**。如下所示：

<div *id=*"app01">
<input *type*="text" *name*="age" *v-bind:value*=**"age + 3**">
</div>
<script>var *vm=* new Vue({
el:'#app01',
data:{
age:12
}
});
</script>

效果为：



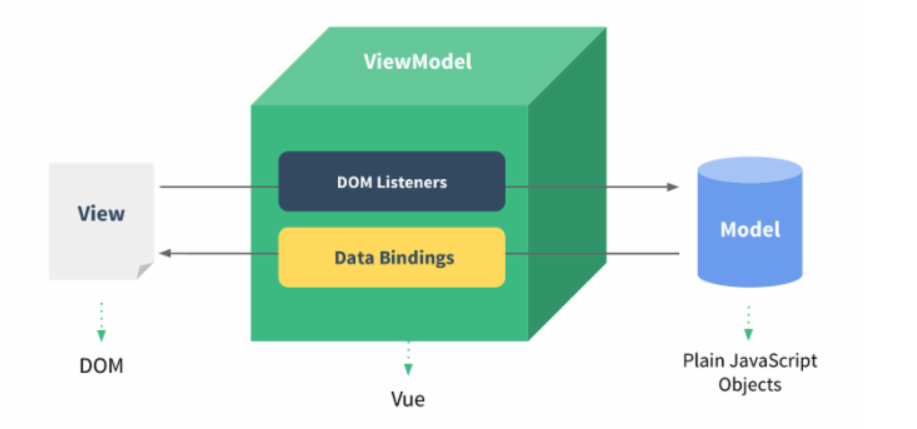
很显然是把age变量的值12取出来，又加上一个3，进行计算得到的。

当然**v-bind:** 也可以简写为 **:[冒号]**，如下所示：



**v-model**

v-model:唯一一个实现双向数据绑定的命令



<div id="app01">
<div>{{age}}</div>
<!-- **v-bind:只能实现数据的单项绑定，只能从M绑定到V中，无法实现数据的双向绑定** -->
<input type="name" v-bind:value="age">
<!-- **只有v-model命令是实现 表单元素（View）和 Model 数据的 双向数据绑定** -->
<input type="age" v-model="age">
</div>
<script>
var vm= new Vue({
el:'#app01',
data:{
age:12
}
});
</script>

注意：**v-model只能运用在表单元素中**，所谓表单元素是指：

**select、input(text、email、adress、radio,checkbox）、textarea**

**事件绑定**

可以用v-on指令监听DOM事件，并在触发时运行一些JavaScript代码

**用于绑定事件 v-on : 事件 = “函数名” 简写 @事件=“函数名”**

**v-on:click**

<body>
<div id="app">
{{message}}
<button v-on:click="fun1">Vue的OnClick</button>
</div>
<script>
 **new Vue({**
 **el:"#app", //由Vue接管id为app的区域**
 **data:{**
 **message:"Hello Vue" //注意：此处不要加分号**
 **},**
 **methods:{ //这个methods属性中定义都是vm实例所有可用的方法**
 **fun1:function(){**
 **alert("您好啊");**
 **}**
 **}**
 **});**
</script>
</body>

当然此函数也是可以接受参数的，如下所示：

<body>
<div id="app">
{{message}}
<button v-on:click="fun1('Vue on click')">Vue的OnClick</button>
</div>
<script>
new Vue({
el:"#app", //由Vue接管id为app的区域
data:{
message:"Hello Vue" //注意：此处不要加分号
},
methods:{
fun1:function(msg){
alert("您好啊");
this.message = msg;
}
}
});
</script>
</body>

所以：**v-bind是绑定属性的，**

**v-on是绑定事件的。**



注意：在vm实例中，要想获取data中的数据或者methods中的方法，必须通过this.属性名或者this.方法名的方式调用，这里的this就表示我们new 出来的VM实例对象。

**v-on:mouseover**

将v-on:click换成v-on:mouseover也是可以的

<div *id=*"app01">
<button *v-on:mouseover*="func('皮吧你')" >点我啊</button>
</div>
**v-on:blur**
<div *id=*"app01">
<input *v-on:blur*="func('皮吧你')" />
</div>

**v-on:keydown**

**鼠标点击到按钮，然后键盘按下，就会弹出这个窗口**。

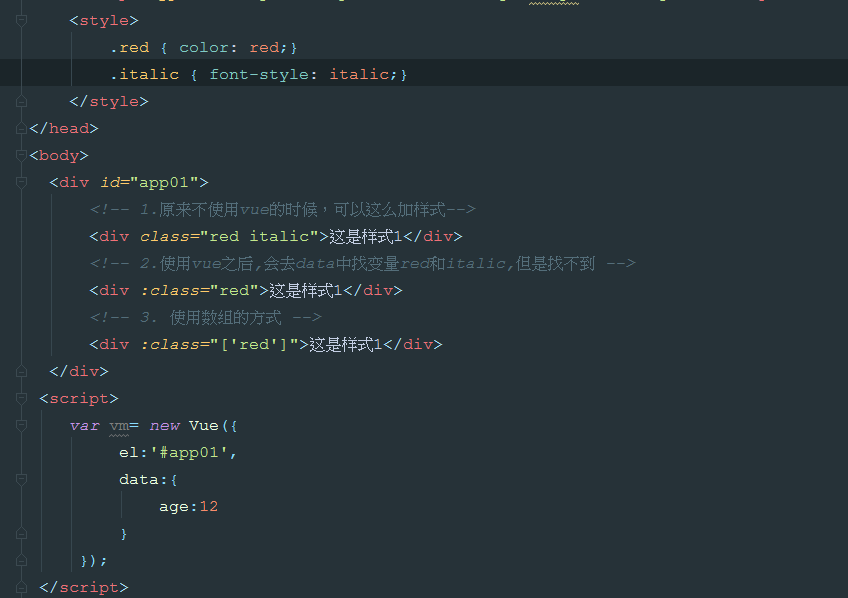
<div *id=*"app01">
<button *v-on:keydown*="func('皮吧你')" >点我啊</button>
</div>

**Vue中使用样式**

Vue的样式正常情况下有点多余，使用原生的class或者style样式即可，但是在某些逻辑下是有用的，例如当一点击的时候就变色。

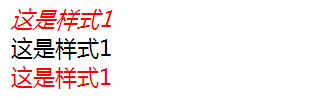
**使用class的样式**

**使用数组**

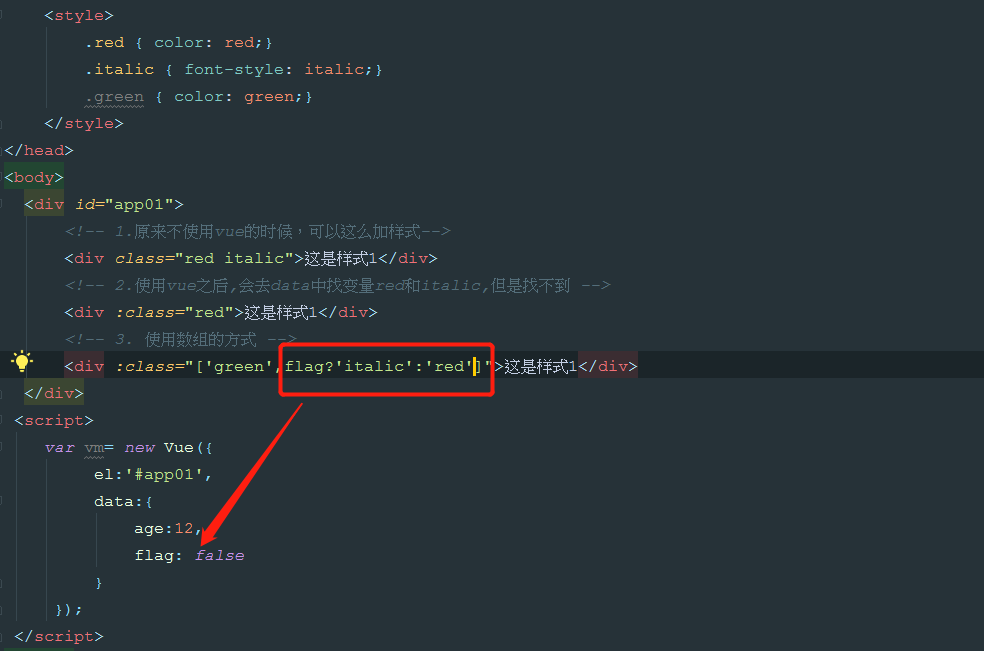


**注意：使用数组的时候，必须将red加上单引号，否则，会作为变量去vm实例的data中去找数据。而且必须使用v-bind命令。**

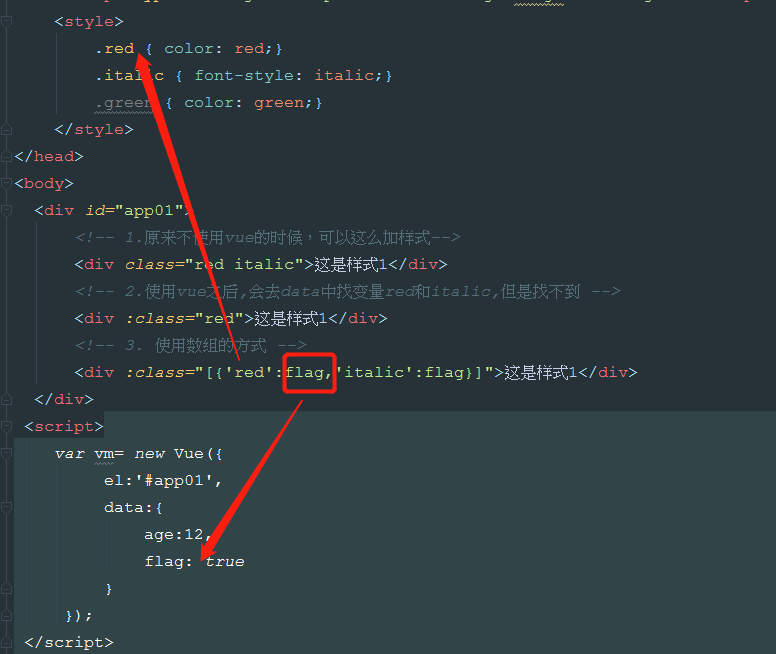
效果图：



**在数组中使用三元表达式**



**数组中使用json对象**

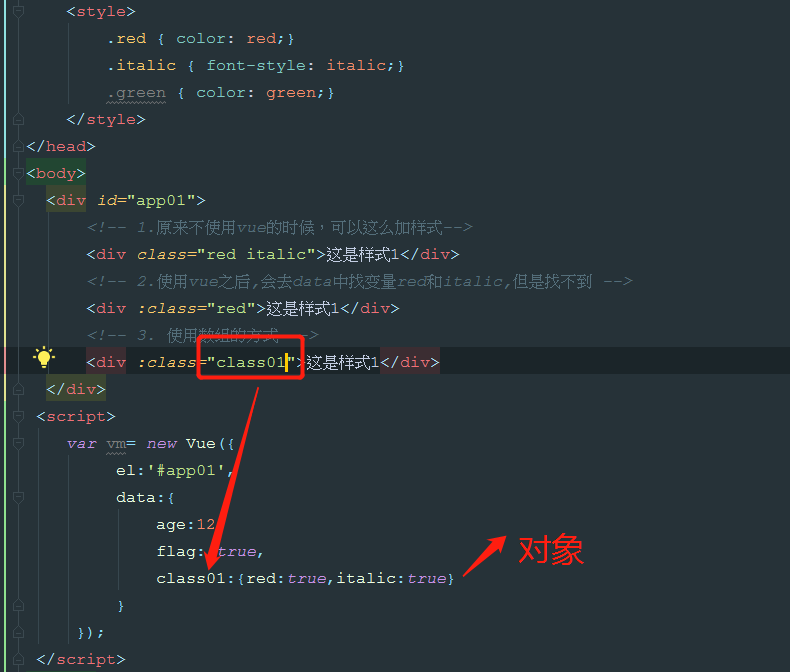


**4.不使用数组，直接使用对象**



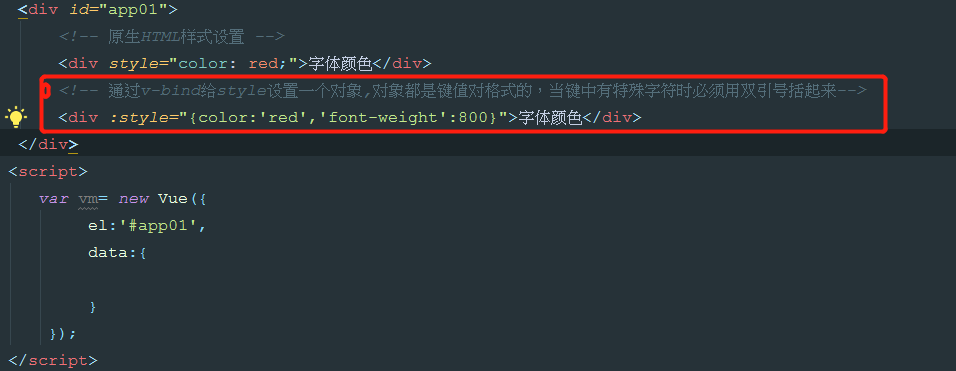
**这时候，json对象中的key虽然引用的是上边的样式，确不用再加单引号,当然也可以加引号。**

再变：

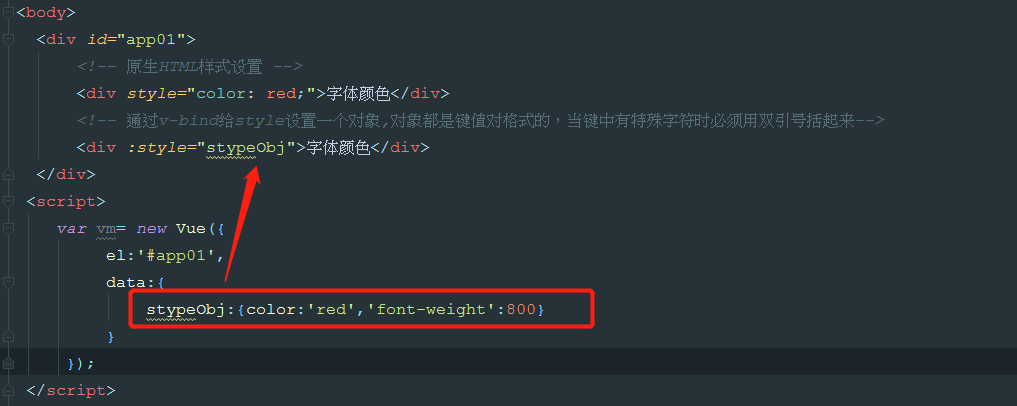


**使用内联样式**

style里面直接使用json对象,如下所示：



变1：将样式对象放到data中。

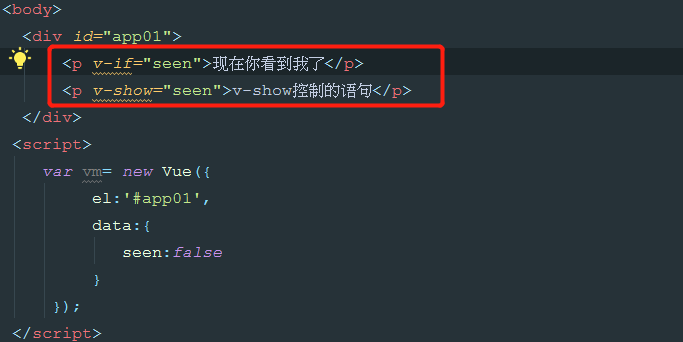


再变：v-bind命令后面跟一个样式对象数组



**Vue的结构**

**v-if语句**

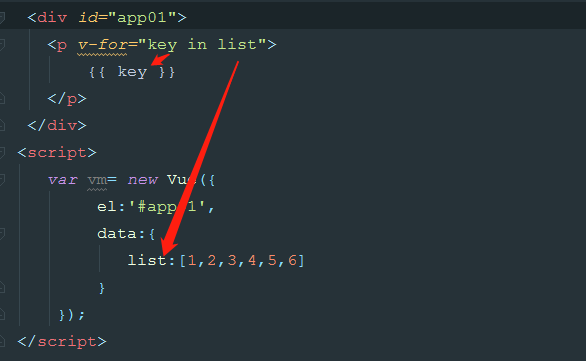


v-if和v-show是一样的，这是在隐藏的时候，v-show是通过给标签加display:none的样式完成的，而v-if是直接将元素去掉完成的。

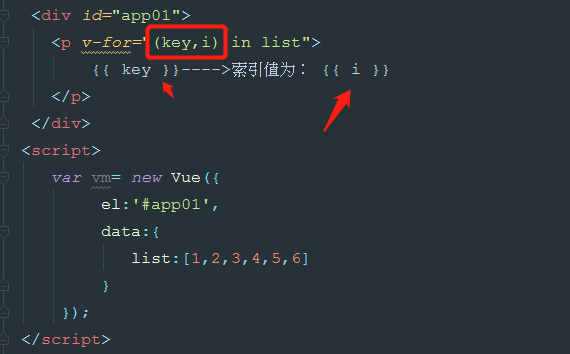


**v-for指令**

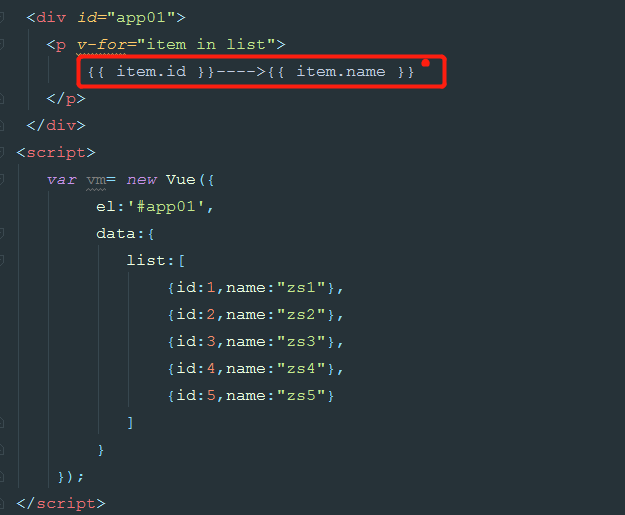
**遍历数组**



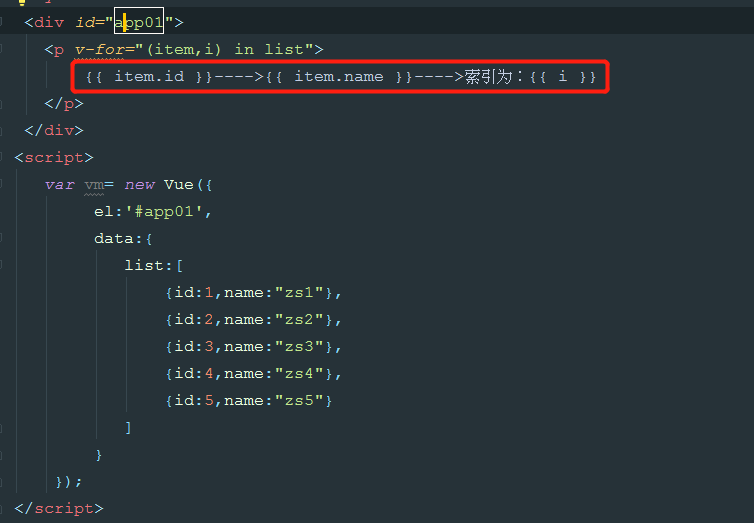
如果需要索引值，则



再变：循环对象数组



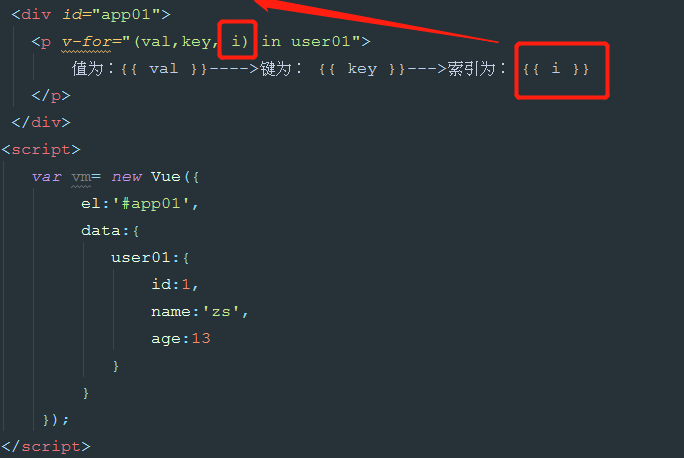
同样，想要索引，也可以加上：



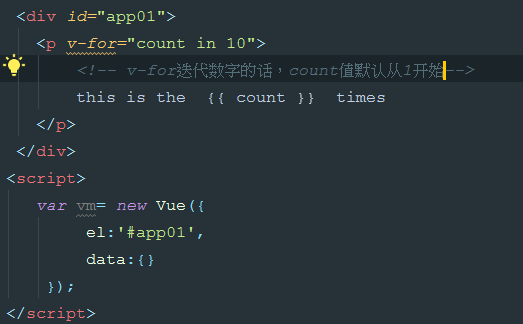
**遍历对象**



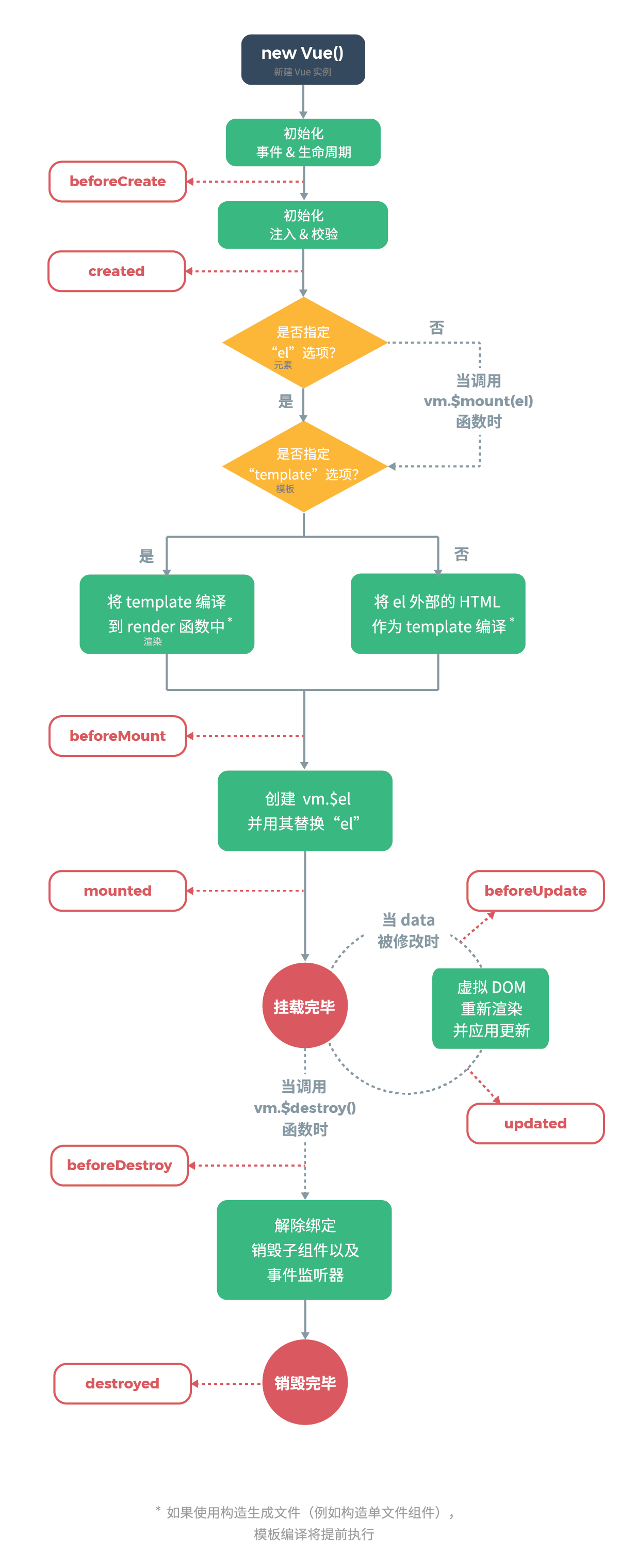
在遍历对象身上的键值对的时候，除了，有val、key之外，在第三个位置还有 一个索引。



**遍历数字**



**vue的生命周期**



**Vue实现异步交互**

Vue可以通过axios可以进行异步交互。

参考：<https://www.runoob.com/vue2/vuejs-ajax-axios.html>

<head>
<meta *charset*="UTF-8">
<title>Title</title>
<scri*pt ty*pe="text/javascrip*t" s*rc="static/js/vuejs-2.5.16.js"></script>
<scr*ipt t*ype="text/javascri*pt"* src="static/js/axios.js"></script>
</head>
<body>
 *<*div id="app01">
 *<table* borde*r="1px" cell*padding*="10px" cell*spacing="0px">
 *<*tr v-for="item *in in*fo" :key="item.eid">
<td>{{ item.eid }}</td>
<td>{{ item.ename }}</td>
<td>{{ item.esex }}</td>
<td>{{ item.eage }}</td>
</tr>
</table>
</d*iv>* <sc*ript*>
var vm= new Vue({
el:'#app01',
data:{
info: ""
},
mo*unte*d () {
 var \_this = this;
axios.get*('/pms/e*mp/test01').then(function(result) {
\_this.info = result.data;
});
}
});
</script>

更进一步：使用element-ui组件显示table表格数据

<head>
<meta *charset*="UTF-8">
<title>Title</title>
<scri*pt ty*pe="text/javascrip*t" s*rc="static/js/vuejs-2.5.16.js"></script>
<scr*ipt t*ype="text/javascri*pt"* src="static/js/axios.js"></script>*<!-- 引入样式 --*>
*<lin*k rel="styles*heet"* href="https://unpkg.com/element-ui/lib/theme-chalk/index.css*">
<!-- 引入组件库* -->
*<scr*ipt src="https://unpkg.com/element-ui/lib/index.js"></script>
</head>
<b*ody*>
<div id="app01">
<div >
<el-table
 :data="info"
 style="width: 50% *;margin*:auto*;" :stri*pe=true :border="true" >
<el-table-column
 align="center"
 prop="eid"
 label="EID"
 width="180">
</el-table-column>
<el-table-colum*n* align="cente*r"* prop="ena*me"* label=*"姓名"* width="180">
</el-table-column>
<el-table*-colum*n
align=*"cent*er"
pr*op="ea*ge"
label="年齡">
</el-table-column>
</el-table*>* *</di*v>
</div>
<script>
var vm= new Vue({
el:'#app01',
data:{
info: []
 },
 mounted () {
var \_this = this;
 axios.get('/pms/emp/test01').then(function(result) {
\_this.info = result.data;
});
}
});
</script>

**Vue实现分页**

<head>
<meta *charset*="UTF-8">
<title>Title</title>
<scri*pt ty*pe="text/javascrip*t" s*rc="static/js/vuejs-2.5.16.js"></script>
<scr*ipt t*ype="text/javascri*pt"* src="static/js/axios.js"></script>*<!-- 引入样式 --*>
*<lin*k rel="styles*heet"* href="https://unpkg.com/element-ui/lib/theme-chalk/index.css*">
<!-- 引入组件库* -->
*<scr*ipt src="https://unpkg.com/element-ui/lib/index.js"></script>
</head>
<bo*dy>*
<div id="app01">
<div >
<el-table
 :data="info"
 style="width: 50% *;margin:*auto;*" :strip*e=true :border="true" >
<el-table-column
 align="center"
 prop="eid"
 label="EID"
 width="180">
</el-table-column>
<el-table-column align="center*"* prop="enam*e"* label="*姓名"* width="180">
</el-table-column>
<el-table-*column*
align="*cente*r"
pro*p="eag*e"
label="年齡">
</el-table-column>
</el-table>
 *<el-pagination*background
layout="*prev, p*ager, next"
:*total="tota*lRecord"
 *:page-size=pa*geSize
*:current-page=pa*geNum
 @current-change="currentChange" style="float:right;margin: 10px 335px 0 0;">
</el-pag*inat*ion> </div>
</div>
<script>
var vm= new Vue({
el:'#app01',
data:{
info: [],
pageNum:1,
pageSize:3,
totalRecord:100
 *},* methods: {
*curr*entChange: function (currentPage) {
 this.pageNum=currentPage;
this.request()
 },
 request:function () {
var \_this = this;
 *axio*s.get('/pms/emp/test01?pageNum='+\_this.pageNum).then(function(result) {
\_this.info = result.d*ata;* }); }
},
mounted:function() {
this.request();
}
});
</script>