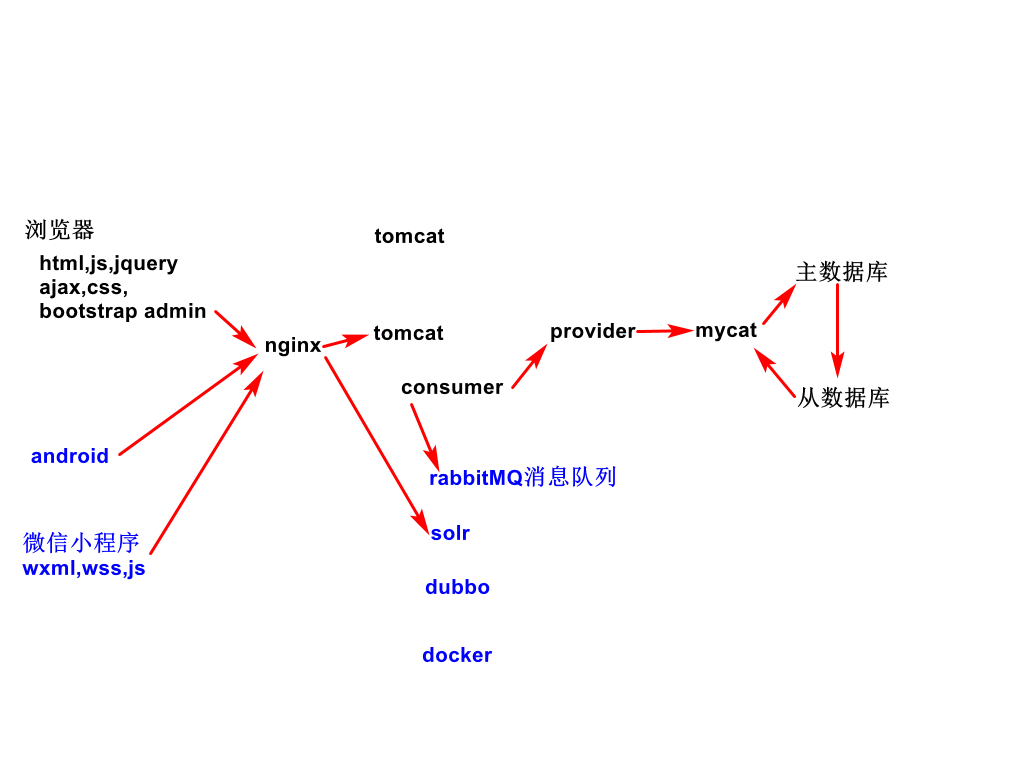
# 联系方式

张久军 13260051510

添加微信时写明 1810中心名称 姓名



# 课程安排



# Bootstrap

## 拼多商城演示

演示web端，android移动端，小程序端。

## 定义

Bootstrap是基于HTML、CSS、JavaScript 开发的简洁、直观、强悍的前端开发框架，使得 Web 开发更加快捷。 提供了优雅的HTML和CSS规范，能制作响应式网页。

## Bootstrap使用场景

运行intel项目

运行“亿发,移动端课前资料\bootstrap\WebContent\intel完整版”文件夹中的index.html

## Bootstrap 原理

### 媒体查询

创建static web project项目 bootstrap。

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<style type=*"text/css"*>

/\*\*样式在屏幕宽度>0 <768的设备上生效\*\*/

@media only screen and (min-width: 0px) and (max-width: 768px) {

**body** {

background-color: *red*;

}

}

/\*\*样式在屏幕宽度>=768 <992的设备上生效\*\*/

@media only screen and (min-width: 768px) and (max-width: 992px) {

**body** {

background-color: *green*;

}

}

/\*\*样式在屏幕宽度>=992的设备上生效\*\*/

@media only screen and (min-width: 992px) {

**body** {

background-color: *blue*;

}

}

</style>

</head>

<body>

</body>

</html>

### 组合样式

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<style type=*"text/css"*>

*.item* {

color: *red*;

}

*.title* {

font-size: *80px*;

}

</style>

</head>

<body>

<div>test1</div>

<div class=*"item"*>test2</div>

<div class=*"title"*>test3</div>

<div class=*"item title"*>test4</div>

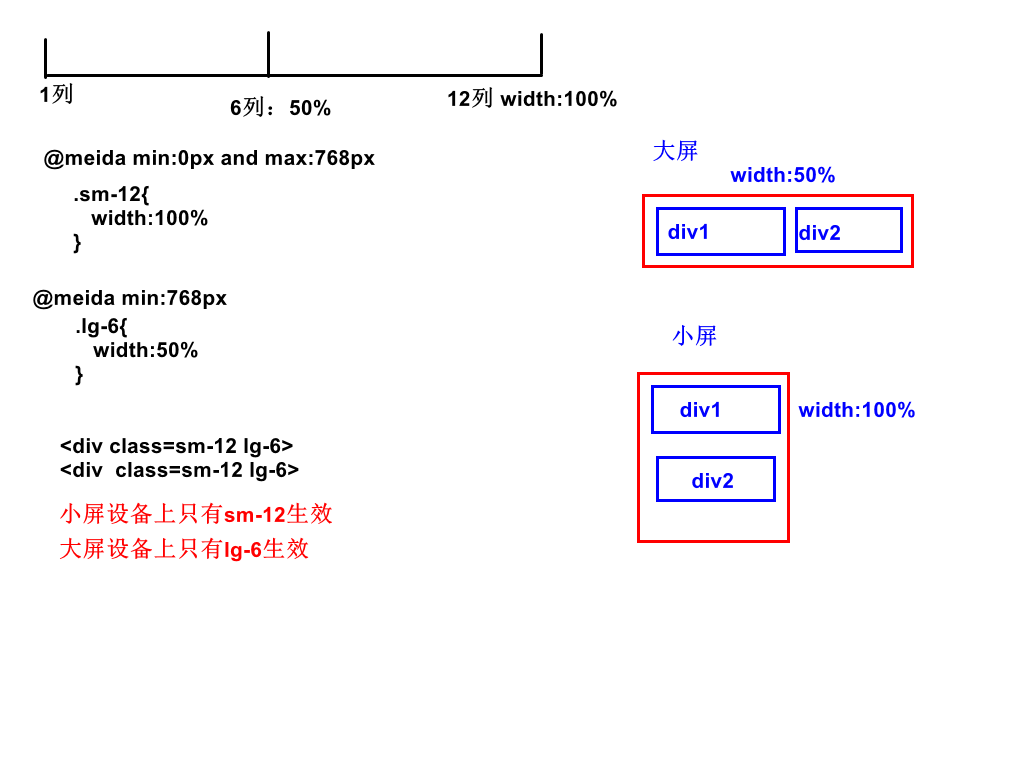
</body>

</html>

### 不同屏幕显示的网页不一样

\培优课前资料\亿发,移动端课前资料\bootstrap\WebContent\03mediaquery.html

代码分析



<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<style type=*"text/css"*>

**div** {

float: *left*;

}

/\*\*屏幕分成12列，宽度是12列 width=100% 宽度是6列 width=50%\*\*/

/\*\*屏幕宽度在0到768之间\*\*/

@media only screen and (min-width: 0px) and (max-width: 768px) {

*.sm-12* {

width: *100%*;

}

}

/\*\*屏幕宽度大于768\*\*/

@media only screen and (min-width: 768px){

*.lg-6* {

width: *50%*;

}

}

</style>

</head>

<body>

<!--

sm-12是100% lg-6是50%

两个会同时生效吗？

在0-768 sm-12 生效

在>768 设备上lg-6生效

-->

<div class=*"sm-12 lg-6"*>div1</div>

<div class=*" sm-12 lg-6"*>div2</div>

</body>

</html>

在chrome浏览器，elements窗口中会发现当屏幕大于768时,lg-6生效。当宽度小于768时，只有sm-12生效。

## 屏幕分类

Bootstrap根据屏幕大小，把设备分成超小屏幕，小屏幕，中等屏幕，大屏幕。

|  |  |  |  |
| --- | --- | --- | --- |
| 超小屏幕  手机 0-768px | 小屏幕  平板768-992 | 中等屏幕  电脑992-1200 | 大屏幕  电脑1200-? |
| Col-xs- | Col-sm- | Col-md- | Col-lg- |

在Bootstrap源码1625行中定义了col-xs-12样式，如图-1

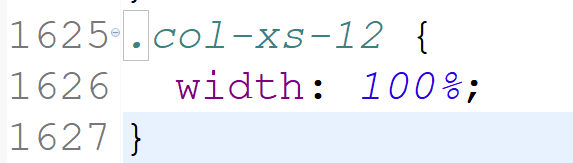


图- 1

## Bootstrap使用

使用bootstrap框架制作一个网页，网页标签嵌套关系统如图-2

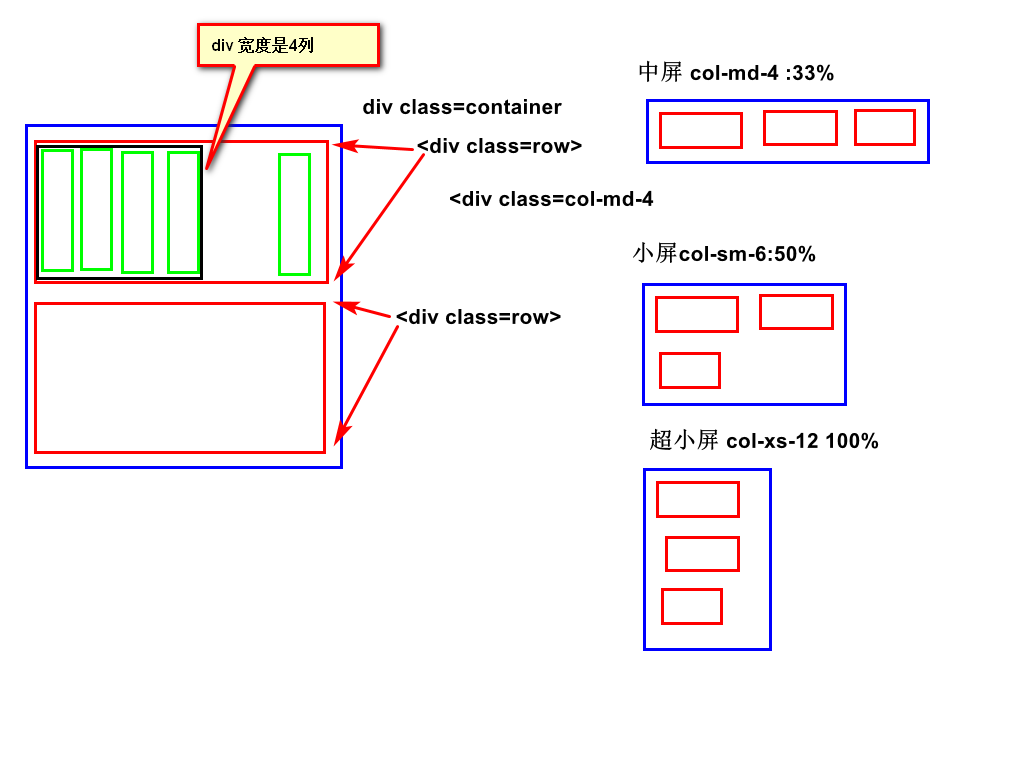


图- 2

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title></title>

<link rel=*"stylesheet"* type=*"text/css"* href=*"../css/bootstrap.css"*/>

</head>

<body>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-xs-12 col-sm-6 col-md-4"*>1111</div>

<div class=*"col-xs-12 col-sm-6 col-md-4"*>222</div>

<div class=*"col-xs-12 col-sm-6 col-md-4"*>333</div>

</div>

</div>

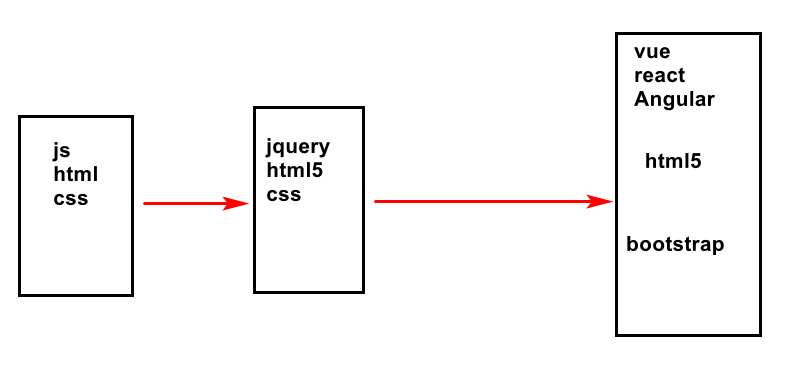
</body>

</html>

在chrome浏览器，elements窗口中会发现当屏幕宽度小于768时，只有com-xs-12生效。

# Vue

## 前端技术介绍



## Vue介绍

是个最新的前端框架，比jquery,ajax开发效率高，功能强大。

www.iviewui.com-->开始使用-->指南 查看那些公司在使用vue.

https://www.iviewui.com/docs/guide/introduce

## 手写Vue框架部分功能

### 根元素

#### 需求

自动管理控件

#### 分析

1. 传统做法

用jquery找到根元素，设置背景色

用框架

架构 师写框架

1，定义配置文件json中的key options{el:rootElement}

2，js中写类Vue,从配置文件el key找根元素，设置背景色

-----------------------------------------

程序员做项目时使用框架

1. 写配置信息 options{el:rootElement}

2，启动框架

#### 代码

创建static web project项目vue

1. html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<div id=*"app"*>内容不能为空

</div>

</body>

<script type=*"text/javascript"*>

//架构师写vue框架

//定义配置文件的内容 对象中必须有el key,el放的是根元素的id

//定义类 options是配置信息

**function** Vue(options)

{

//从配置信息中找根元素

**var** rootElementId=options.el;

**var** divObject=document

.getElementById(rootElementId);

//设置背景色

divObject.style.backgroundColor="#FF0000";

}

//------------------------------

//程序员使用框架

//1,设置配置信息

**var** options={

"el":"app"

}

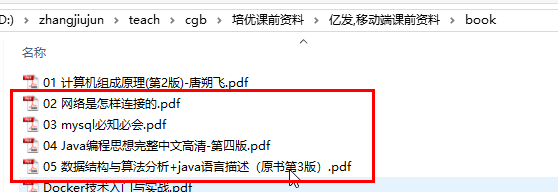
//2,启动框架

**var** vue=**new** Vue(options);

</script>

</html>

# 需要看的书



# 手写Vue框架部分功能

### 数据绑定

#### 需求

从框架中取数据自动显示在网页上

#### 分析

Jquery 实现方法

1. 找到根元素root,
2. 创建一个元素span,
3. root.append(span)

架构师写框架

1.从options.data中取数据

2.Vue()中显示数据

程序员做项目用框架

1. 在options中申明data
2. 调用vue()函数

#### 代码

02.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<div id=*"app"*></div>

</body>

<script type=*"text/javascript"*>

//1,架构师写框架

**function** Vue(options) {

//options={el:app,data:{item:{},user:{}}}

**var** el = options.el;

**var** rootElement = document.getElementById(el);

//给根元素赋值

**var** title = options.data.item.title;

rootElement.innerHTML = title;

}

//----------------------

//2,程序员使用框架

//1,设置配置信息

**var** options = {

el : "app",

data : {

item : {

title : "手机"

},

user : {}

}

}

//2,启动框架

**var** vue = **new** Vue(options);

</script>

</html>

### Mounted事件

#### 需求

框架在网页加载完成后，自动执行一个函数

Jquery实现：$.document.ready()

Js实现：Window.onLoad=function

#### 分析

架构师写框架

1.自动调mounted函数

程序员做项目用框架

1.实现mounted函数

#### 代码

03.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

</body>

<script type=*"text/javascript"*>

//框架

//1,定义配置文件格式和内容

/\*\*

\* options={

mounted:fun

}

\*/

//2,根据配置，去执行函数

**function** Vue(options) {

//mounted是一个函数

**var** mounted = options.mounted;

//网页加载完运行函数

window.onload = mounted;

}

//---------------------------------------

//程序员

//1,设置配置信息

**var** options = {

mounted : **function**() {

//chrome浏览器 F12 console窗口

console.log("run");

//实际工作中，写很多代码

}

}

//2,启动框架

**var** vue = **new** Vue(options);

</script>

</html>

### 实时自动更新数据

#### 需求

添加数据后，自动实时更新数据

用jquery实现

Insert(){

创建元素span

找到根元素root

Root.append(span)

}

#### 分析

架构师写框架

1.每隔10ms从options.data中取数据，显示在option.el指定的根元素上。

程序员做项目用框架

1. Body中增加 div,insert Btn标签
2. 创建options.data
3. 实现insert()

#### 代码

04.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<!-- 数据绑定 -->

<button onClick="insert()">insert</button>

<div id=*"app"*></div>

</body>

<script type=*"text/javascript"*>

//1,框架

**function** Vue(options) {

//1,找到根元素

**var** el = options.el;

**var** rootElement = document.getElementById(el);

//2,从配置对象中取数据

**var** items = options.data.items;

//3,自动更新界面

**function** run() {

console.log("test");

//清除根元素上的内容

rootElement.innerHTML = "";

//把items数组中的数据显示在根元素中

**for** (**var** i = 0; i < items.length; i++) {

//var a=a+1;

rootElement.innerHTML = rootElement.innerHTML + items[i].title

+ "<br/>";

}

}

//每隔1秒钟执行一次run,相当于java中的thread

window.setInterval(run, 1000);

}

//-------------------

//2,程序员

**var** options = {

el : "app",

data : {

items : [ {

title : "item1"

}, {

title : "item2"

} ]

}

}

**var** vue = **new** Vue(options);

**function** insert() {

//把新数据放到数组中，vue框架会让界面会自动更新

**var** item = {

title : "新商品"

};

//items[0] [1]

//items[lenght]放新商品

**var** length = options.data.items.length;

//添加的新商品放在数组的最后面。

options.data.items[length] = item;

}

</script>

</html>

### 事件绑定

#### 需求

框架实现事件绑定，程序员不用处理事件绑定

#### 分析

架构师写框架

[1.框架规定通过@click](mailto:1.框架规定通过@click)设置事件

2.遍历网页中所有元素，判断元素有没有加事件，如果加事件了，自动从options找一个函数来处理

程序员做项目用框架

1.<button @click=”deleteTest”>

2.options{deleteTest:function()}

#### 代码

05.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<!-- @click属性是由vue框架来处理

传统做法 onClick=deleteTest()

-->

<button @click=*"deleteTest"*>删除</button>

</body>

<script type=*"text/javascript"*>

//1,框架

**function** Vue(options) {

//1,找到所有的button

**var** buttons = document.getElementsByTagName("button");

**for** (**var** i = 0; i < buttons.length; i++) {

**var** button = buttons[i];

//一个标签有多个属性

**var** attributes = button.attributes;

//遍历属性数组

**for** (**var** k = 0; k < attributes.length; k++) {

**var** attribute = attributes[k];

//属性名

**var** name = attribute.name;

//2,判断button有没有@click属性

**if** (name == "@click") {

//3,从配置对象中找一个函数，进行事件绑定

button.onclick = options.deleteTest;

}

}

}

}

//------------------

//2,程序员使用框架

**var** options = {

deleteTest : **function**() {

console.log("删除");

}

}

**var** vue = **new** Vue(options);

</script>

</html>

### 灵活事件绑定

#### 需求

调用用户指定的事件

#### 分析

架构师写框架

1，判断程序员写的函数是不是与@click的值一致，是 ，设置事件处理函数

程序员做项目用框架

1. 写deleteTest(),modify()函数
2. 为删除按钮，修改按钮设置事件处理函数

#### 代码



06.html

jquery-3.1.1.min.js在 \亿发,移动端课前资料\vue文件夹中

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<script src=*"jquery-3.1.1.min.js"* type=*"text/javascript"*>

</script>

</head>

<body>

<!-- @click属性是由vue框架来处理

传统做法 onClick=deleteTest()

-->

<button @click=*"deleteTest"*>删除</button>

<button @click=*"modify"*>修改</button>

</body>

<script type=*"text/javascript"*>

//1,框架

**function** Vue(options) {

//1,找到所有的button

**var** buttons = document.getElementsByTagName("button");

**for** (**var** i = 0; i < buttons.length; i++) {

**var** button = buttons[i];

//一个标签有多个属性

**var** attributes = button.attributes;

//遍历属性数组

**for** (**var** k = 0; k < attributes.length; k++) {

**var** attribute = attributes[k];

//属性名

**var** name = attribute.name;

//2,判断button有没有@click属性

**if** (name == "@click") {

//3,从配置对象中找一个函数，进行事件绑定

/\*\*

\*<button @click=deleteTest

\*<button @click=modify>

\*options{

\* deleteTest:fun,

\* modify:fun

\*}

\*/

//button @click的属性值 deleteTest,modify

**var** attributeValue = attribute.value;

//jquery 的each函数遍历options,取到每个数据后，都执行function(key,value)

//取到第一个数据时key就是deleteTest,value是我们定义的函数fun(){}

$.each(options, **function**(key, value) {

**if** (attributeValue == key) {

button.onclick = value;

}

});

}

}

}

}

//------------------

//2,程序员使用框架

//jquery $.each()遍历options对象

//key deleteTest=第一个按钮的@click属性值

//key modify=第二个按钮的@click属性值 button.onclick=

**var** options = {

deleteTest : **function**() {

console.log("删除");

},

modify : **function**() {

console.log("修改");

}

}

**var** vue = **new** Vue(options);

</script>

</html>

### 综合案例

#### 需求

用户添加的数据，自动显示在网页上，能删除数据

#### 分析

架构师写框架

1.数据绑定

2.事件绑定

程序员做项目用框架

1. div,insertBtn,deleteBtn
2. options{el,data,insert函数,delete函数}

#### 代码

07.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<script src=*"jquery-3.1.1.min.js"* type=*"text/javascript"*>

</script>

</head>

<body>

<button @click=*"insert"*>insert</button>

<button @click=*"deleteTest"*>delete</button>

<div id=*app*></div>

</body>

<script type=*"text/javascript"*>

//1写框架

**function** Vue(options) {

//1，找到根元素

**var** el = options.el;

**var** rootElement = document.getElementById(el);

//2,数据绑定

**function** run() {

**var** items = options.data.items;

rootElement.innerHTML = "";

**for** (**var** i = 0; i < items.length; i++) {

**var** title = items[i].title;

rootElement.innerHTML = rootElement.innerHTML + title + "<br/>"

}

}

window.setInterval(run, 100);

//3,事件绑定

//3.1 找到所有的button

**var** buttons = document.getElementsByTagName("button");

**for** (**var** i = 0; i < buttons.length; i++) {

**var** button = buttons[i];

**var** attributes = button.attributes;

**for** (**var** k = 0; k < attributes.length; k++) {

**var** name = attributes[k].name;

//3.2 判断button有没有@click属性

**if** ("@click" == name) {

//3.3 判断options.key与@click属性值是否一样

**var** attributeValue = attributes[k].value;

/\*\*

options{

el:

data:

deleteTest:fun

insert:fun

}

\*/

$.each(options, **function**(key, value) {

**if** (attributeValue == key) {

button.onclick = value;

}

})

}

}

}

}

//-------------------------------------

//2程序员用框架

**var** options = {

el : "app",

data : {

items : []

},

insert : **function**() {

**var** item = {

title : "新商品"

};

//向数组中添加一个数据

options.data.items.push(item);

},

deleteTest : **function**() {

//从数组中移出一个数据

options.data.items.pop();

}

}

**var** vue = **new** Vue(options);

</script>

</html>