

# Vue第二天

---

## 今日内容介绍

---

- v-on按键修饰符
- 扩展品牌管理需求来学习新知识点
- Vue中的AJAX请求
- 跨域复习和JSONP
- Vue的生命周期方法
- 实现品牌管理案例的AJAX版

## 今日内容学习目标

---

- 掌握v-on事件按键修饰符的作用
- 记住自定义属性指令和元素指令的写法
- 记住自定义过滤器的写法以及管道符|的使用
- 能够分清楚私有过滤器和全局过滤器的应用场景
- 掌握vue-resource中get方法的使用
- 掌握vue-resource中jsonp方法的使用
- 掌握vue-resource中post方法的使用

## 详细内容

---

### 1 v-on按键修饰符

---

#### 1.1 作用说明

文档地址: <http://cn.vuejs.org/v2/guide/events.html#按键修饰符>

在监听键盘事件时，我们经常需要监测常见的键值。Vue 允许为 v-on 在监听键盘事件时添加按键修饰符：

```
.enter  
.tab  
.delete (捕获“删除”和“退格”键)  
.esc  
.space  
.up  
.down  
.left  
.right  
1.0.8+ 支持单字母按键别名。
```

#### 1.2 可以自定义按键别名

在Vue2.0 中默认的按键修饰符是存储在 `Vue.config.keyCodes`中  
// 例如在Vue2.0版本中扩展一个f1的按键修饰符写法：  
`Vue.config.keyCodes.f1 = 112`

在1.0.17+ 中默认的按键修饰符是存储在`Vue.directive('on').keyCodes`中

// 例如在Vue1.0中扩展一个f1的按键修饰符写法：  
`Vue.directive('on').keyCodes.f1 = 112`

### 1.3 利用v-on的.enter按键修饰符实现品牌管理的新增按钮功能

- 效果图

3

奔驰

添加品牌

当光标在这个文本框中按回车键自动添加数据功能

| 编号 | 品牌名称 | 创建时间                                       | 操作                 |
|----|------|--|--------------------|
| 1  | 宝马   | Thu Mar 09 2017 18:36:27 GMT+0800 (中国标准时间) | <a href="#">删除</a> |
| 2  | 奥迪   | Thu Mar 09 2017 18:36:27 GMT+0800 (中国标准时间) | <a href="#">删除</a> |

- 实现代码

```

<script src="../../vue1028.js"></script>
</head>
<body>
  <div id="app">
    <div>
      <input type="text" v-model="productid" >
      <input type="text" v-model="productname" @keydown.enter="addProduct" >
      <button @click="addProduct">添加品牌</button>
    </div>
  </div>
  ...
  new Vue({
    el: '#app',
    data: {
      list: [
        {id: 1, name: '宝马', ctime: new Date()},
        {id: 2, name: '奥迪', ctime: new Date()}
      ],
      productid: 0,
      productname: '',
      searchValue: '' //代表搜索文本框中的内容,通过v-model就能够自动同步到视图中的数据
    },
    methods: {
      // 2.0 添加
      addProduct: function(e) {
        // if(e.keyCode !== 13){
        //   return;
        // }
        // alert(this.productid + " ,"+this.productname);
        // 1.0 获取到页面上的数据
        var pobj = {id: this.productid, name: this.productname, ctime: new Date()};
        // 2.0 添加数据
        this.list.push(pobj);

        // 3.0 清空模型productid和productname的值
        this.productname = '';
        this.productid = 0;
      }
    }
  });

```

## 2 自定义指令

当Vue提供的系统指令不能满足需求时，就需要自己定义指令来进行扩展，例如，定义一个v-focus指令来实现文本框的自动获取焦点功能

### 2.1 自定义属性指令

- 写法格式

定义指令：

```
Vue.directive('指令ID, 不需要增加v-前缀',function(){  
  //实现指令的业务  
  this.el //代表使用这个指令的元素对象  
});
```

使用指令(当做一个元素的属性使用)：

```
<input type="text" v-指令ID />
```

- （属性指令应用举例）利用自定义属性指令实现自动获取焦点功能

定义指令：

```
//定义一个 v-focus的属性自定义指令  
Vue.directive('focus',function(){  
  this.el.focus(); //实现文本框的自动获取焦点  
});
```

使用指令：

```
<input type="text" v-focus />
```

## 2.2 自定义元素指令

- 写法格式

定义指令：

```
Vue.elementDirective('指令id',{  
  bind:function(){  
    //实现指令的业务  
    this.el //代表使用这个指令的元素对象  
  }  
});
```

使用指令：

```
<指令id></指令id>
```

- （元素指令应用举例）利用自定义属性指令实现日期格式化

定义指令：

```
Vue.directive('datefmt',{
  bind:function(){
    var v=this.el.attributes[0].value;
    var date = new Date(this.vm[v]);
    var year = date.getFullYear();
    var m = date.getMonth() + 1;
    var d = date.getDate();
    //输出: yyyy-mm-dd
    var fmtStr = year+'-'+m+'-'+d;

    this.el.innerText = fmtStr;
  }
});

new Vue({
  el:'#app',
  data:{
    time:new Date()
  }
});
```

使用指令：

```
<div id="app">
  <datefmt :dt="time"></datefmt>
</div>
```

## 3 过滤器

Vue提供了一系列的固定逻辑来使程序员更加容易的实现这些功能，这些过滤器称之为系统过滤器，Vue也提供了一个接口用来供程序员定义属于自己的特殊逻辑，Vue称之为自定义过滤器

### 3.1 系统过滤器

- 关于系统过滤器的使用参考请参考文档：<http://v1-cn.vuejs.org/api/#过滤器>
- 注意：系统过滤器是Vue1.0中存在的，在Vue2.0中已经删除了

### 3.2 自定义过滤器

- 文档地址：<http://v1-cn.vuejs.org/guide/custom-filter.html>

#### 3.2.1 自定义私有过滤器

- 定义方式

可以在 `new Vue({filters: {}})` 中的 `filters` 中注册一个私有过滤器

定义格式:

```
new Vue({
  el: '#app',
  filters: {
    '过滤器名称': function(管道符号 | 左边参数的值, 参数1, 参数2, ...) {
      return 对管道符号 | 左边参数的值做处理以后的值
    }
  }
});
```

Vue1.0 使用写法:

```
<span>{{ msg | 过滤器id '参数1' '参数2' .... }}</span>
```

Vue2.0 使用写法:

```
<span>{{ msg | 过滤器id('参数1' '参数2' ....) }}</span>
```

- (应用示例)自定义全局过滤器实现日期格式化

1、定义全局的日期格式化过滤器:

```
new Vue({
  el: '#app',
  data: {
    time: new Date()
  },
  filters: {
    // 定义在 VM 中的 filters 对象中的所有过滤器都是私有过滤器
    datefmt: function(input, splicchar) {
      var date = new Date(input);
      var year = date.getFullYear();
      var m = date.getMonth() + 1;
      var d = date.getDate();
      var fmtStr = year + splicchar + m + splicchar + d;
      return fmtStr; // 返回输出结果
    }
  }
});
```

2、使用

```
<div id="app">
  {{ time | datefmt '-' }} //Vue1.0传参写法

  {{ time | datefmt('-',) }} //Vue2.0传参写法

</div>
```

### 3.2.2 自定义全局过滤器

- 定义方式

可以用全局方法 `Vue.filter()` 注册一个全局自定义过滤器，它接收两个参数：过滤器 ID 和过滤器函数。过滤器函数以值为参数，返回转换后的值

定义格式：

```
Vue.filter('过滤器名称', function (管道符号|左边参数的值,其他参数1,其他参数2,...) {  
    return 对管道符号|左边参数的值做处理以后的值  
})
```

Vue1.0 使用：

```
<span>{{ msg | 过滤器名称 '参数1' '参数2' .... }}</span>
```

Vue2.0 使用：

```
<span>{{ msg | 过滤器名称('参数1' '参数2' ....) }}</span>
```

- (应用示例)自定义全局过滤器实现日期格式化

- 1、 定义全局的日期格式化过滤器：

```
Vue.filter('datefmt',function(input,splicchar){  
    var date = new Date(input);  
    var year = date.getFullYear();  
    var m = date.getMonth() + 1;  
    var d = date.getDate();  
    var fmtStr = year+splicchar+m +splicchar+d;  
    return fmtStr; //返回输出结果  
});
```

- 2、使用

```
<div id="app">  
    {{ time | datefmt '-' }} //Vue1.0传参写法  
  
    {{ time | datefmt('-') }} //Vue2.0传参写法  
  
</div>  
<script>  
    new Vue({  
        el:'#app1',  
        data:{  
            time:new Date()  
        }  
    });  
</script>
```

## 4 Vue中的AJAX请求

Vue可以借助于vue-resource来实现AJAX请求

- http请求报文

浏览器与服务器数据交互是遵循http协议的，当浏览器要访问服务器的时候，浏览器需要将相关请求数据提交给服务器（例如：浏览器信息，url地址，参数等），通常是通过请求报文来提交的

请求报文的格式分为：

- 1、请求报文行
- 2、请求报文头
- 3、请求报文体

- http响应报文

当浏览器请求服务器的时候，服务器需要将数据返回给浏览器，这种数据是通过响应报文响应回浏览器的

响应报文的格式分为：

- 1、响应报文行
- 2、响应报文头
- 3、响应报文体

- 请求报文和响应报文配图

The screenshot shows the 'Headers' tab of a web browser's developer tools. The top section displays the request details, and the bottom section displays the response details. Red annotations highlight specific parts of the request and response.

**Request Details:**

- Request Line:** GET /build.js HTTP/1.1 (highlighted with a red box and labeled '请求报文行')
- Request Headers:** Host: localhost:8099, Connection: keep-alive, User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36, Accept: \*/\*, Referer: http://localhost:8099/, Accept-Encoding: gzip, deflate, sdch, br, Accept-Language: zh-CN, zh;q=0.8, en;q=0.6, If-None-Match: "3ae4fbe08089d21:0", If-Modified-Since: Sat, 18 Feb 2017 00:49:37 GMT (highlighted with a red box and labeled '请求报文头').
- Request Body:** uname=ivan (highlighted with a red box and labeled '请求报文体, post请求用来向服务器传参数').

**Response Details:**

- Response Line:** HTTP/1.1 304 Not Modified (highlighted with a red box and labeled '响应报文:行').
- Response Headers:** Accept-Ranges: bytes, ETag: "3ae4fbe08089d21:0", Server: Microsoft-IIS/8.5, X-Powered-By: ASP.NET, Date: Mon, 13 Mar 2017 08:05:34 GMT (highlighted with a red box and labeled '响应报文头').
- Response Body:** <script>alert('ok')</script> (highlighted with a red box and labeled '响应报文体').

## 4.1 vue-resource

Vue与后台Api进行交互通常是利用vue-resource来实现的，本质上vue-resource是通过http来完成AJAX请求响应的

- vue-resource GitHub 地址: <https://github.com/pagekit/vue-resource>



- vue-resource Http请求api参考（主要看这个）：<https://github.com/pagekit/vue-resource/blob/master/docs/http.md>
- vue结合vue-resource写法步骤

1、通过 <https://cdn.jsdelivr.net/vue.resource/1.2.1/vue-resource.min.js> 下载到vue-resource文件

2、在html页面中通过script标签导入vue-resource.min.js 文件后，就会自动的在Vue对象实例上初始化\$http

3、使用

// 全局Vue对象写法

```
Vue.http.get('/someUrl', [options]).then(successCallback, errorCallback);  
Vue.http.post('/someUrl', [body], [options]).then(successCallback, errorCallback);
```

// 在Vue对象中的写法

```
this.$http.get('/someUrl', [options]).then(successCallback, errorCallback);  
this.$http.post('/someUrl', [body], [options]).then(successCallback, errorCallback);
```

- vue-resource get请求

写法格式：

```
this.$http.get('请求的url', [可选参数对象, 使用{}传参]).then(成功回调函数, 失败回调函数);
```

成功回调函数参数对象主要属性说明：

- 1、url：请求的原始url
- 2、body：响应报文体中的数据（我们通常用这个属性获取服务器返回的数据）
- 3、其他属性请看文档

举例：

```
this.$http.get('http://vuecms.ittun.com/api/getlunbo?id=1').then(function(res)  
{console.log(res.body)}, function(err){//err是异常数据});
```

- vue-resource post请求

写法格式:

```
this.$http.post('请求的url',[可选参数请求报文体对象body,使用{}传参],[可选参数对象,使用{}传参]).then(成功回调函数,失败回调函数);
```

成功回调函数参数对象主要属性说明:

- 1、url : 请求的原始url
- 2、body: 响应报文体中的数据(我们通常用这个属性获取服务器返回的数据)
- 3、其他属性请看文档

注意点:

`$http.post()`方法中的第二个参数固定写成: `{emulateJSON:true}`, 否则可能造成服务器无法接收到请求报文体中的参数值

举例:

```
this.$http.post('http://vuecms.ittun.com/api/adddata?id=1' //请求的url
,{content:'hello'} //请求报文体中传入的参数对象, 多个使用逗号分隔
,{emulateJSON:true} //固定写法, 保证服务器可以获取到请求报文体参数值
).then(function(res){console.log(res.body)}, function(err){//err是异常数据});
```

- vue-resource jsonp请求

jsonp请求主要用来解决ajax跨域请求问题, 使用jsonp实现跨域首先要保证服务器api支持jsonp请求的格式

写法格式:

```
this.$http.jsonp('请求的url',[可选参数对象,使用{}传参]).then(成功回调函数,失败回调函数);
```

成功回调函数参数对象主要属性说明:

- 1、url : 请求的原始url
- 2、body: 响应报文体中的数据(我们通常用这个属性获取服务器返回的数据)
- 3、其他属性请看文档

举例:

```
this.$http.jsonp('http://vuecms.ittun.com/api/getlunbo?id=1').then(function(res)
{console.log(res.body)}, function(err){//err是异常数据});
```

## 4.2 利用vue-resource完成品牌管理案例的AJAX版本

```
<script src="../../vue1028.js"></script>
<script src="../../vue-resource.js"></script>
```

```
methods:{
  //1.0 删除
  del:function(id){
    //向http://vuecms.ittun.com/api/delproduct/:id发送ajax 的get请求
    //1.0 定义url
    var url = 'http://vuecms.ittun.com/api/delproduct/'+id;

    //2.0 发出请求
    this.$http.get(url).then(function(res){
      alert(res.body.message);
    });

    //3.0 刷新页面
    this.getlist();
  },
  del2:function(index){
    this.list.splice(index,1);
  },

  // 2.0 添加
  addProduct:function(e){
    //通过ajax的post请求来增加数据
    //1.0 确定url
    var url = 'http://vuecms.ittun.com/api/addproduct'
    //2.0 post(url,请求报文体的数据,{emulateJSON:true}).then()

    this.$http.post(url,{name:this.productname},{emulateJSON:true}).then(function(res){
      // alert(res.body.message);
    });

    //3.0 重新获取列表数据
    this.getlist();
  },
  //3.0 从服务器获取到品牌数据
  getlist:function(){
    this.$http.get('http://vuecms.ittun.com/api/getprodlist')
    .then(function(res){
      if(res.body.status!==0){
        alert(res.body.message);
        return;
      }

      //正常处理
      this.list = res.body.message;
    });
  }
}
};
```

## 5 Vue的生命周期方法

## Vue1.0版本

## Vue2.0版本

