# AJAX

## 1、简介

AJAX：（Asynchronnous JavaScript And XML）异步的JavaScript与XML。

AJAX 不是新的编程语言，而是一种使用现有标准的新方法。

Ajax不是一种新技术，而是几种蓬勃发展的几种强大技术以新的方式组合而成，Ajax包括：基于XHTML和CSS标准的表示；使用Document Object Model进行动态的显示和交互；使用XML Request与服务器进行异步通讯；使用JavaScript绑定一切。

Ajax的工作原理，Ajax的核心是JavaScript和XMLHttpRequest。该对象在Internet Explorer5中首次引入，它是一种支持异步请求的技术。简单的说：XmlHttpRequest使你可以使用JavaScript向服务器提出请求并处理响应，而不阻塞用户。在创建Web站点时，在客户端执行屏幕更新为用户提供了很大的灵活性能。下面是使用Ajax完成的功能：1、动态更新购物车的物品总数，无需用户单击Update并等待服务器发送整个页面。2、提升站点的性能，这是通过减少从服务器下载的数据量来实现的。例如：当更新购物车一个物品的数量时，会重新载入整个页面，这必须下载32K的数据，而使用Ajax计算新的总量，服务器只需要返回新的总量值，因此所需要的带宽仅仅为原来的百分之一。3、消除了用户每次输入时的页面刷新。例如，在Ajax中，如果用户在分页列表上单击Next，则服务器数据只刷新列表而不是整个页面。

## 2、XMLHttpRequest/ActiveXObject

XMLHttpRequest 对象，现代浏览器均支持 XMLHttpRequest 对象（IE5 和 IE6 使用 ActiveXObject）。

XMLHttpRequest 用于在后台与服务器交换数据。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

创建 XMLHttpRequest 对象，所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera）均内建 XMLHttpRequest 对象。创建 XMLHttpRequest 对象的语法：

var xhr =new XMLHttpRequest();

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject ：

var xmlhttp;

if (window. ActiveXObject){

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}else{

xmlhttp=new XMLHttpRequest();

}

向服务器发送请求的方式为：

xmlhttp.open(method,URL,true);

举例：

xmlhttp.open(“GET”,”./ajaxServlet”,true);

xmlhttp.open(“POST”,”./ajaxServlet”,true);

## 3、实例讲解

### 1）解析xml信息

从Ajax的序列的起点开始，创建和分派来自浏览器的XMLHttpRequest。但不同的浏览器创建XMLHttpRequest的方法各不相同。下列JavaScript的代码，可以检测出浏览器要使用的正确方法，并返回一个可以使用的XMLHttpRequest。

<script language="JavaScript">

<!--

//定义xmlHttp全局对象

**var xmlHttp = null;**

//根据不同浏览器创建方法的不同进行区分,确定xmlHttp对象的顺利创建

**function queryMess(){**

**if(window.ActiveXObject){**

**xmlHttp = new window.ActiveXObject("Microsoft.XMLHTTP");**

**}else if(window.XMLHttpRequest){**

**xmlHttp = new window.XMLHttpRequest();**

**}**

//事先给xmlHttp对象获取数据返回后一个处理数据的函数，即：回调函数

//xmlHttp.onreadystatechange属性，是xmlHttp对象的状态改变所触发事件的事件处理程序

**xmlHttp.onreadystatechange = dealDate；**

//调用xmlHttp.open()方法来设置请求后台服务器资源及请求方式

//第一个参数表示以何种方式来进行请求，值为两个“GET”或者“POST”

//第二个参数表示请求的服务器资源，举例“./testServlet”表示请求的这个Servlet的url资源

**xmlHttp.open("GET",url,true);**

//xmlHttp.send(null);用来发送信息参数，如果是以GET的方式进行请求，那么send()或者send(null);

//如果是以POST方式请求，那么必须设置请求的格式，如下行代码所示

//XMLHttpReq.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

//send()方法中必须填写要发送的参数信息

/\*\*

一个参数的情况

**xmlHttp.send("province="+province);**

**setRequestHeader("Content-Type","application/x-www-form-urlencoded ");**

两个参数的情况

**xmlHttp.send(“value1=”+value1+”&value2=”+value2);**

\*/

}

//定义一个回调函数，用作xmlHttp对象获取数据后进行处理的操作

**function dealDate(){**

/\*xmlHttp.readyState属性，是xmlHttp对象的对象状态值，

0 描述一种"未初始化"状态；此时，已经创建一个XMLHttpRequest对象，但是还没有初始化。   
1 描述一种"未发送"状态；此时，代码已经调用了XMLHttpRequest open()方法并且XMLHttpRequest已经准备好把一个请求发送到服务器。   
2 描述一种"发送"状态；此时，已经通过send()方法把一个请求发送到服务器端，但是还没有收到一个响应。   
3 描述一种"正在接收"状态；此时，已经接收到HTTP响应头部信息，但是消息体部分还没有完全接收结束。   
4 描述一种"已加载"状态；此时，响应已经被完全接收。

\*/

**if(xmlHttp.readyState == 4){**

/\*xmlHttp.status从服务器返回的数字代码，404表示未找到，200表示就绪，**不管是否获取**

**到数据，200表示已经顺利完成访问服务器后，顺利返回未发生异常**

\*/

**if(xmlHttp.status == 200){**

/\*数据返回后可以使用xmlHttp.responseText属性获取数据，但是xmlHttp.responseText

返回的是一个字符串，对于比较复杂的数据结构可以利用xmlHttp.responseXML属性来进行

处理数据,如果返回值是一条记录，则可以处理数据的XML样式为：

<?xml version="1.0" encoding="gb2312" ?>

<result>根节点

<username>sa</username>子节点

<userid>23</userid>子节点

</result>

可以根据节电名称来获取节点值，例如：\*/

var responseXML = xmlHttp.reponseXML;

var username = responseXML.getElementsByTagName("username")[0].firstChild.nodeValue;

var userid = responseXML.getElementByTagName("userid")[0].firstChild.nodeValue;

/\*

若返回值为一组数据，则可以处理数据的XML样式为：

<? xml version="1.0" encoding="UTF-8" ?>

<students>根节点

<student>子节点

<name>sa</name>

<age>12</age>

</ student >

< student >

<name>ss</name>

<age>22</age>

</ student >

</ students >对XML进行解析来获取数据

\*/

//获取根节点，获取根节点的节点名称：var domName = responseXML.documentElement.tagName;

var xmlDom = xmlHttpRequest.responseXML.documentElement;

for(var i=0;i<xmlDom.childNodes.length;i++){

var student = xmlDom.childNodes.item(i);

var name = student.childNodes.item(0).textContent;

var age = student.childNodes.item(1).textContent;

alert(name);alert(age);

}

}else{

alert("获取数据发生错误!");

}

}

//-->

</script>

### 2）生成菜单树

### 3）二级联动

JavaScript代码

var xmlHttpRequest = null;

function proviceChange(){

var province = document.getElementById("province").value;

if(window.ActiveXObject){

xmlHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");

document.getElementById("ieType").innerHTML="Microsoft";

}else{

xmlHttpRequest = new XMLHttpRequest();

document.getElementById("ieType").innerHTML="Firefox";

}

xmlHttpRequest.open("POST","./TestServlet", true);

xmlHttpRequest.onreadystatechange = changeCity;

xmlHttpRequest.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xmlHttpRequest.send("province="+province);

}

//Firefox浏览器下正常执行

function changeCity(){

if(xmlHttpRequest.readyState==4){

if(xmlHttpRequest.status==200){

var xmlDom = xmlHttpRequest.responseXML.documentElement;

var length = xmlDom.childNodes.length;

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

var city = xmlDom.childNodes.item(i);

var name = city.childNodes.item(0).textContent;

var code = city.childNodes.item(1).textContent;

var citySelect =document.getElementById("city");

citySelect[i+1] = new Option(name,code);

}

}

}

}

//考虑浏览器兼容问题，使用原始的Dom方式解析XML，IE和Firefox下均可执行。

function changeCity(){

if(xmlHttpRequest.readyState==4){

if(xmlHttpRequest.status==200){

//对于xml的解析，IE浏览器与Firefox并不相同

var xmlDom = null;

if(window.ActiveXObject){

var xmlObj = new ActiveXObject("Microsoft.XMLDOM");

xmlObj.async = false;

xmlObj.loadXML(xmlHttpRequest.responseText);

xmlDom = xmlObj.getElementsByTagName("citys")[0];

}else{

xmlDom = xmlHttpRequest.responseXML.documentElement;

}

var length = xmlDom.childNodes.length;

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

var city = xmlDom.childNodes.item(i);

var name = city.childNodes.item(0).firstChild.nodeValue;

var code = city.childNodes.item(1).firstChild.nodeValue;

var citySelect =document.getElementById("city");

citySelect[i+1] = new Option(name,code);

}

}

}

}

Servlet代码

String provCode = req.getParameter("province");

PrintWriter out = resp.getWriter();

String xml = "<?xml version='1.0' ?><citys>";

if("37".equals(provCode)){

xml = xml + "<city><name>济南</name><code>0531</code></city>";

xml = xml + "<city><name>青岛</name><code>0532</code></city>";

xml = xml + "<city><name>淄博</name><code>0533</code></city>";

}

if("13".equals(provCode)){

xml = xml + "<city><name>石家庄</name><code>1301</code></city>";

xml = xml + "<city><name>唐山</name><code>1302</code></city>";

xml = xml + "<city><name>秦皇岛</name><code>1303</code></city>";

}

xml = xml + "</citys>";

out.print(xml);

out.close();

# 二、JQuery

## 1、简介

JQuery就是简单的javaScript的查询，其中查询是Jquery是强大功能点之一，但是jquery的突出特色不只是查询这一方面，比如在UI设计方面，同样提供了很强大的功能。Jquery的作用就是：write less,do more。写的更少，做的更多。

## 2、语法格式

$(jsObject/”页面内容”).functionName()/attribute

使用Jquery必须熟悉他的语法格式，最突出的特点就是前面带有$符号，后面()中内容为JS的对象，或者在页面中所使用的html标签的tagName或者标签的属性，根据属性获取标签对象使用的方式不同，后面会逐一讲解。

举例：$(document).ready();类似于document.onload();但有不同

$(document).ready(function(){

alert(“hello”);

});

$(document).ready(function(){

alert(“world”);

});

页面弹出内容：hello world

window.onload=function(){

alert("bbb");

}

window.onload=function(){

alert("ccc");

}

页面弹出内容：ccc

## 3、对象选择

Jquery选择对象的方式，所有的选择方式都是基于Jquery的选择器进行，后面会对选择器进行详细讲解。

举例：1、通过ID方式来获取对象

Html页面代码

<div id=”ch1”>aaaa</div><span id=”ch1”>bbbb</span>

$(document).ready(function(){

$(“#ch1”).html(“ssss”);

alert($(“#ch1”).length);

});

结果：只有div标签的内容发生了更改，span标签的内容没有变化，同样弹出对话框信息获取的对象的长度是1。即使获取一个对象也是通过数组集合的方式进行返回。

举例：2、通过tagName标签名称的方式来获取对象

Html代码

<a href=”#”>aaa</a><a href=”#”>bbb</a><a href=”#”>ccc</a><a href=”#”>ddd</a>

$(document).ready(function(){

$(“a”).html(“hhhh”);

alert($(“a”).length);

})

结果：a标签中所有的文本值全部发生变化，通过此种情况我们可以看出Jquery获取的对象是一个集合对象时，它会遍历操作集合中的每个子对象。特殊情况，来获取集合对象中的文本的时候，Jquery并不知道我们具体要获取那个子对象的标签内容，所以只会返回给我们第一个子对象的标签内容。

例如：

$(document).ready(function(){

alert($(“a”).html());

})

弹出信息只会是第一个标签的文本信息：aaa

html()方式是Jquery的定义方法，不带参数获取标签的内容，带参数改变标签的内容。

## 4、JS对象与Jquery对象的相互转换

1）JS对象转换为Jquery对象

举例：1

document对象转为Jquery对象 $(document)

举例：2

<a id=”a1” href=”#”>aaa</a>

var a1 = document.getElementById(“a1”);

var $a2 = $(a1);

$a2 就是Jquery的对象。

2）Jquery对象转换为JS对象

举例：1

<a href=”#”>aaa</a><a href=”#”>bbb</a><a href=”#”>ccc</a><a href=”#”>ddd</a>

var a1 = $(“a”)[0] 或者使用jQuery提供的get方法来获取如： var a1 = $(“a”).get(0);

获取文本时就要用alert(a1.innerHTML);而不能用html()方法。

## 5、Jquery的方法

### 1）html()

<div id=”d1”>abc</div>

$(“#d1”).html();根据ID来获取层所包含的HTML信息内容。

$(“#d1”).html(“hhhhhh”);传入参数来改变层中的HTML信息。

### 2）click()

<a id=”a1” href=”#”>点击触发事件</a>

$(“#a1”).click(function(){

alert(“被点击了”);

})

### 3）css(“”)

<span id=”s1”>行内标签span</span>

$(“#s1”).css(“color”,”red”) 设置写入CSS样式，两个参数

$(“#s1”).css(“color”)获取样式值，一个参数

Jquery的提供的方法很多，就不在进行一一举例，但是所有的方法基本上都是根据元素的属性来进行定义的，提供的方法既可以对属性赋值，也可以获取属性值的信息。后面遇到一个讲一个。

## 6、Jquery选择器（重点）

$()函数在很多javascript类库中都被作为一个选择器函数来使用，在jQuery中同样也使用它来作为选择函数。其中$(“#id”)用来代替document.getElementById(“id”)；$(“a”)用来代替document.getElementsByTagName()函数，即通过标签名获取HTML元素。

jQuery选择器的分类：

1）基本选择器（basic）

2）层次选择器（level）

3）过滤选择器（filter）

4）表单选择器（form）

### 1）基本选择器

$(“#id”)根据ID来匹配一个元素，举例：$(“#test”)表示选取id为test的元素

$(“.test”)根据给定的类名来匹配元素，举例：$(“.test”)表示选取所有class为test的元素

$(“p”)根据给定的元素名类匹配元素，举例：$(“p”)表示选取所有元素名为p的元素

$(“\*”)匹配所有元素，表示选取当前页面所有的元素。

$(“selector1,selector2,selector3”)将每一个选择器匹配到元素合并后一起返回，举例：$(“div,#test,p.myClass”)选取所有元素名为div的页面对象，选取id为test的第一个元素对象，选取p标签元素下，所有class名为myClass的子元素对象，所有的元素组成一个集合进行返回。

### 2）层次选择器

$(“div span”)，选取所有div中的后代元素为span的元素集合（常用的选择器），例如：

<div>

<span>111</span>

<a href=”#”><span>222</span></a>

</div>

层div中的两个span元素都会被选中，虽然第二个span标签不是div层的子元素，但是是它的后代元素，所以同样会被选中。

$(“div > span”)，选取所有div中的为span子元素，例如：上面例子中

<div>

<span>111</span>

<a href=”#”><span>222</span></a>

</div>

只有第一个span元素会被选中。

$(“.class +span”)，选取样式名为class的元素的下一个元素，提示：并不是获取子元素，而是获取的是同级的下一个元素。举例： $(“.test + span”)选中的就是div的同级的下一个span元素

<div class=”test”>

<span>abc</span>

</div>

<span>bbbb</span>

红颜色的span元素对象会被选中，如果样式名为test的元素的下一个元素不为span类型的元素，则选择器将不进行选择，例如：

<div class=”test”>

<span>abc</span>

</div>

<p>1111111</p>

<span>bbbb</span>

那么含有“bbbb”的span元素将不会被选中。

等价于$(“.test”).next(“span”)

$(“#test ~ span”)，同上，获取的id为test元素的后面所有兄弟元素名为span的同级元素。同级下前面的元素并不获取，举例：

<span>1111</span>

<div id=”test”>hhhh</div>

<span>2222</span>

<div>333</div>

<span>4444</span>

第一个span元素并不获取，获取id为test元素的后面所有span元素。

等价于$(“#test”).nextAll(“span”)

使用$(“.class + span”)或者$(“#id +div”)带加号的选择器或者$(“.class ~ div”)或者$(“#id ~ span”)都是选则后面的元素，如果想要前面的元素也选中使用jQuery提供的方法siblings ()可以进行选中，举例：

<span>111111</span>

<p class=”test”>标题行</p>

<span>22222</span>

$(“.test”).siblings(“span”).css(“color”,”red”);

### 3）过滤选择器

* 基本过滤
* 内容过滤
* 可见性过滤
* 属性过滤
* 子元素过滤
* 表单对象属性过滤

#### 基本过滤器

使用一些关键字来进行过滤，如下

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **描述** | **返回** | **示例** |
| **:first** | 选取第一个元素 | 单个元素 | $(“div:first”)选取所有div元素中的第一个div |
| **:last** | 选取最后一个元素 | 单个元素 | $(“div:last”)选取所有div元素中的最后一个div |
| **:not(selector)** | 去除所有与给定选择器匹配的元素 | 集合元素 | $(“input:not(.myClass)”)选取class不是myClass的<input>元素 |
| **:even** | 选取索引是偶数的所有元素，索引从0开始 | 集合元素 | $(“input:even”)选取索引是偶数的<input>元素 |
| **:odd** | 选取索引是奇数的所有元素，索引从0开始 | 集合元素 | $(“input:odd”)选取索引是奇数的<input>元素 |
| **:eq(index)** | 选取索引等于index的元素，索引从0开始 | 单个元素 | $(“input:eq(1)”)选取索引是1的<input>元素 |
| **:gt(index)** | 选取索引大于index的元素，索引从0开始 | 集合元素 | $(“input:gt(1)”)选取索引大于1的所有<input>元素 |
| **:lt(index)** | 选取索引小于index的元素，索引从0开始 | 集合元素 | $(“input:lt(4)”)选取索引值小于4的所有<input>元素 |
| **:header** | 选取所有的标题元素，例如h1，h2，h3等等 | 集合元素 | $(“:header”)选取网页中所有的<h1><h2><h3>标题元素 |
| **:animated** | 选取当前正在执行动画的所有元素 | 集合元素 | $(“:animated”)选取正在执行动画的div元素。 |

#### 内容过滤器

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **描述** | **返回** | **示例** |
| **:contains(text)** | 选取含有文本内容为”text”的元素 | 集合元素 | $(“div:contains(text)”)选取所有div元素中的含有text文本的元素 |
| **:empty** | 选取不包含子元素或者文本的空元素 | 集合元素 | $(“div:empty”)选取所有div空元素 |
| **:has(selector)** | 选取含有选择器所匹配的元素的元素 | 集合元素 | $(“div:has(span)”)选取含有span元素的div元素，注意：区别于contains这个has中可以是元素名，class名，id但不能是文本 |
| **:parent** | 选取含有子元素或者文本的元素 | 集合元素 | $(“div:parent”)选取含有子元素包括文本的div元素。 |

#### 3、可见性过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **描述** | **返回** | **示例** |
| **:hidden** | 选取所有不可见的元素 | 集合元素 | $(“:hidden”) 选取所有不可见元素，包括<input type=”hidden”>或者<div style=”display:none”>等元素，如果只选择某一种，可以使用  $(“input:hidden”) |
| **:visible** | 选取所有可见的元素 | 集合元素 | $(“div:visible”)选取所有可见的div元素 |

使用show(mins)方法进行显示

举例：$(“div:hidden”).show(1000).css(“color”,”red”);

#### 4、属性过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **描述** | **返回** | **示例** |
| **[attribute]** | 选取拥有此属性的元素 | 集合元素 | $(“div[id]”)选取拥有id属性的div元素 |
| **[attr=value]** | 选取属性值为value的元素 | 集合元素 | $(“div[id=test]”)选取id值等于test的div元素 |
| **[attr!=value]** | 选取属性值不为value的元素 | 集合元素 | $(“div[id!=test]”)选取id值不等于test的div元素，注意没有id属性的div同样会被选中 |
| **[attr^=value]** | 选取属性值以value开始元素 | 集合元素 | $(“div[title^=test]”)选取属性title以test开始的元素 |
| **[attr$=value]** | 选取属性值以value结束的元素 | 集合元素 | $(“div[title$=test]”)选取属性title以test结束的元素 |
| **[attr\*=value]** | 选取属性值含有value的元素 | 集合元素 | $(“div[title\*=test]”)选取属性title含有test的元素 |
| **[selector1][selector2]**  **[selectorn]** | 用属性选择器合并成一个复合属性选择器，满足多个条件。每选择一次范围缩小一次。 | 集合元素 | $(“div[id][title$=test]”)选取含有属性id，并且title属性以test结尾的div元素。 |

#### 5、子元素过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **描述** | **返回** | **示例** |
| **:nth-child(index/even**  **/odd/equation)** | 选取每个父元素下的第index子元素或者奇偶元素(index从1算起)  所有的元素索引都是从1开始 | 集合元素 | :eq(index)只匹配一个元素，而:nth-child将为每一个父元素匹配子元素，并且:nth-child(index)是从1开始的，而:eq(index)是从0开始的 |
| **:first-child** | 选取父元素的第一个子元素 | 集合元素 | :first只返回单个元素，而:first-child选择符将为每个父元素匹配第一个子元素  例如：$(“div span:first-child”)选取每个div下面的第一个span元素 |
| **:last-child** | 选取父元素的最后一个子元素 | 集合元素 | 同样:last只返回最后一个元素，而:last-child为每个父元素匹配最后一个子元素  举例:$(“div span:last-child”)选取每个div下面的最后一个span元素 |
| **:only-child** | 如果某个元素是它父元素中唯一的子元素，那么将会被匹配。如果父元素中含有其他元素，则不会被匹配 | 集合元素 | $(“div span:only-child”) 选取div元素中仅含有span元素的div元素。冒号前面没有空格 |

注意：子元素过滤时，首先找到父元素，从父元素中进行过滤，所以冒号前面必须有空格，先去匹配父元素。

#### 6、表单属性对象过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **描述** | **返回** | **示例** |
| **:enabled** | 选取所有可用元素 | 集合元素 | $(“#form1 :enabled”)选取id为form1的表单中所有可用的元素 |
| **:disabled** | 选取所有不可用元素 | 集合元素 | $(“#form1 :disabled”)选取id为form1的表单中所有不可用的元素 |
| **:checked** | 选取所有被选中的元素 | 集合元素 | $(“input :checked”)选取所有被选中的<input>元素 |
| **:selected** | 选取所有被选中的选项元素（下拉列表） | 集合元素 | $(“select :selected”)选取所有被选中的选项元素。 |
| **:input** | 选取所有的<input><textarea>  <select>和<button>元素 | 集合元素 | $(“ :input ”)选取所有<input>  <textarea><select><button>元素 |
| **:text** | 选取所有的单行文本 | 集合元素 | $(“:text”)选取所有的text元素 |
| **:password** | 选取所有的密码域 | 集合元素 | $(“:password”)选取所有的密码框 |
| **:radio** | 选取所有的单选按钮 | 集合元素 | $(“:radio”)选取所有的单选按钮 |
| **:checkbox** | 选取所有的复选框 | 集合元素 | $(“:checkbox”)选取所有的复选框 |
| **:submit** | 选取所有的提交按钮 | 集合元素 | $(“:submit”)选取所有的提交按钮 |
| **:reset** | 选取所有的重置按钮 | 集合元素 | $(“:reset”)选取所有的重置按钮 |
| **:button** | 选取所有的button按钮 | 集合元素 | $(“:button”)选取所有的button按钮 |
| **:file** | 选取所有的文件上传域 | 集合元素 | $(“:file”)选取所有的文件上传域 |
| **:hidden** | 选取所有的隐藏域 | 集合元素 | $(“:hidden”)选取所有的隐藏域 |

## 7、Jquery操作Dom对象

HTML中常见的DOM操作

查找节点、增加节点、删除节点、复制节点、替换节点、包裹节点。

### 1）修改属性与文本

举例1：通过Jquery来对一个元素对象进行操作，更改它的文本值，对象属性信息

<div id=”d1”><span title=”span的提示信息”>你好</span></div>

<input type=”button” id=”b1” value=”test” />

<script style=”text/javaScript”>

$(function(){

$(“#b1”).click(function(){

//增加层的title提示信息

$(“#d1”).attr(“title”,”增加div层的提示信息”);

//修改span元素的title提示信息

$("#d1 span:first ").css("color","red");

$("#d1 span:first ").attr("title","新的提示信息");

$("#d1 span:first ").text(“welcome”);

});

});

</script>

### 2）添加节点

首先使用JS方式来添加信息：分两种，一种添加html脚本方式，一种添加dom对象方式

<div id=”d1”></div>

创建输入域的个数：<input type=”text” id=”inputNumber” size=”10”><br>

<input type=”button” value=”test” id=”b1” onclick=” createInput()”>

一、使用html脚本方式实现

function createInput(){

var num = document.getElementById("inputNumber").value;

var htmlStr = "";

for(var i=0;i<num;i++){

htmlStr += "<input type='text' size='15' value='' /><br>";

}

document.getElementById("div1").innerHTML=htmlStr;

}

二、使用dom对象方式

function createInput(){

var num = document.getElementById("inputNumber").value;

for(var i=0;i<num;i++){

var inputText = document.createElement("input");

inputText.setAttribute("type","text");

document.getElementById("div1").appendChild(inputText);

var br = document.createElement("br");

document.getElementById("div1").appendChild(br);

}

}

三、使用jQuery方式

首先介绍jQuery添加元素所涉及的方法，跟js有所区别。

|  |  |  |
| --- | --- | --- |
| 方法名 | 说明 | 示例 |
| append() | 向每个匹配的元素内部追加内容 | <span>abc</span>  $(“span”).append(“<b>你好</b>”)  结果：  <span>abc<b>你好</b></span> |
| appendTo() | 将所有匹配的元素追加到指定的元素中。 | <span>abc</span>  $(“<b>你好</b>”).appnedTo(“span”)  结果：  <span>abc<b>你好</b></span> |
| prepend() | 向每个匹配的元素内部前置内容 | <span>abc</span>  $(“span”).prepend(“<b>你好</b>”)  结果：  <span><b>你好</b>abc</span> |
| prependTo() | 将所有匹配的元素前置到指定的元素中。 | <span>abc</span>  $(“<b>你好</b>”).prependTo(“span”)  结果：  <span><b>你好</b>abc</span> |
| after() | 在每个匹配的元素后面插入内容 | <span>abc</span>  $(“span”).after(“<b>你好</b>”)  结果：  <span>abc</span><b>你好</b> |
| insertAfter() | 将所有匹配的元素插入到指定元素的后面。 | <span>abc</span>  $(“<b>你好</b>”).insertAfter(“span”)  结果：  <span>abc</span><b>你好</b> |
| before() | 在每个匹配的元素之前插入内容 | <span>abc</span>  $(“span”).before(“<b>你好</b>”)  结果：  <b>你好</b><span>abc</span> |
| insertBefore() | 将所有匹配的元素插入到指定的元素的前面。 | <span>abc</span>  $(“<b>你好</b>”).insertBefore(“span’)  结果：  <b>你好</b><span>abc</span> |

function createInput(){

var num = document.getElementById("inputNumber").value;

//var htmlStr = “”;

for(var i=0;i<num;i++){

//htmlStr += “<input type=’text’ size=’10’><br>”;

$(“#div1”).append(“<input type=’text’ size=’10’><br>”);

}

}

### 3）移动节点

利用现有的方法进行节点的移动。

举例：

<ul>

<li>111111</li>

<li>222222</li>

<li>333333</li>

<li>444444</li>

<li>555555</li>

</ul>

<ul>

<li>666666</li>

<li>777777</li>

</ul>

$(function(){

//如果有两个ul元素，取第一个ul元素中的子元素进行对象的移动。

$("ul:first li:eq(1)").insertAfter("ul:first li:eq(3)");

})

### 4）删除节点

使用jQuery提供的remove()方法，此方法是将对象从页面中删除，但是有返回对象，返回的就是删除的对象。

此方法还可以接受一个参数，参数是一个选择器对象，含义是将符合选择器的元素进行删除掉。

<ul>

<li title=”1”>1111</li>

<li title=”2”>2222</li>

<li title=”3”>3333</li>

<li title=”4”>4444</li>

</ul>

$(function(){

var $li = $(“ul:first li:eq(1)”).remove();

$li.appendTo(“ul:first”);

$(“ul:first li”).remove(“li[title=3]”);

$("ul:first li").remove("li:eq(1)");

})

与之类似的方法还有empty()方法，此方法是清空元素中的内容，但是元素对象并不在页面中删除。

<ul>

<li title=”1”>1111</li>

<li title=”2”></li>

<li title=”3”>3333</li>

<li title=”4”>4444</li>

</ul>

$(function(){

var $li = $(“ul:first li:eq(1)”).empty();

})

### 5）复制节点

复制节点，就是将对象重复拷贝一份，可以使用jQuery提供的clone()方法，此方法就是将对象本身进行拷贝，拷贝后返回拷贝的对象。此外clone()方法还可以接收一个boolean参数，此参数表示拷贝的对象是否具有模板对象的事件机制。

举例1：

<ul>

<li>111111</li>

<li>222222</li>

<li>333333</li>

<li>444444</li>

</ul>

<script type=”text/javascript”>

$(function(){

$(“ul li”).click(function(){

var $li = $(this).clone();

$(“ul”).append($li);

//或者使用appendTo()方法

$(this).clone().appendTo(“ul”);

/\*\*

以上代码拷贝的对象不在具有click事件，当clone()方法接受参数为true真时，会将模板对象的

事件机制同时拷贝。

\*/

$(this).clone(true).appentTo(“ul”);

});

})

</script>

### 6）替换节点

* replaceWith()

将当前节点对象替换为一个新的节点对象

举例：

<span>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).replaceWith(“<h1>你好</h1>”);

});

</script>

* replaceAll()

使用一个新的节点对象替换已有的节点对象

举例：

<span>abc</span>

<script type=”text/javascript”>

$(“<h1>你好</h1>”).replaceAll(“span”);

</script>

replaceAll表示替换所有的节点对象

### 7）包裹节点

* wrap()

将当前的节点对象使用一个新的对象来进行包裹，也就是用新的对象将当前的节点对象嵌套进来，并且将元素嵌套在最里一层。

举例：

<span>abc</span>

<script type=”text/javascript”>

$(“span”).wrap(“<a href=’http://www.baidu.com’><b></b> </a>”);

</script>

结果：

<a href=<http://www.baidu.com>><b> <span>abc</span></b></a>

* wrapInner()

将当前节点对象的文本使用一个新的对象来进行包裹，同样也就是使用新的节点对象将当前节点的文本嵌套进来，并且将文本嵌套在最里一层。

<span>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).wrapInner(“<a href=’http://www.baidu.com’><b><b> </a>”);

})

</script>

结果：

<span><a href=<http://www.baidu.com>><b> abc</b></a></span>

### 8）属性节点

* attr()方法

设置与获取属性

举例：

<span title=”abc”>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).attr(“title”,”weclome”);

});

</script>

结果：<span title=”weclome”>abc</span>

在页面标签元素中，每个标签元素都拥有多个属性，因此jQuery也提供了使用属性对象的方式来对元素节点进行操作，属性对象JS中的面向对象方式：{“title”:”weclome”,”id”:”sp1”}，这个方式同时也是JSON所使用的传输方式，也就是JSON串。也可以使用Object对象方式来使用。

<span title=”abc”>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).attr({“title”:”weclome”,”id”:”sp1”});

});

</script>

结果：<span title=”weclome” id=”sp1”>abc</span>

var obj = new Object();

obj.title = “welcome”;

obj.id=”sp1”;

$(“span”).attr(obj);

* removeAttr()

移除元素属性

举例：

<span title=”sp1”>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).removeAttr(“title”);

})

</script>

结果：<span>abc</span>

### 9）遍历节点

children()方法，用来获取当前元素的子节点对象。

next()，获取下一个元素节点。

prev()，获取前一个元素节点。

siblings()，获取同级兄弟元素节点。

<div id=”d1”>

<ul>

<li>1111</li>

<li>2222</li>

<li>3333</li>

</ul>

<span>abc</span>

<p>abc</p>

</div>

<script type=”text/javascript”>

$(function(){

var $children = $(“#d1”).children();

alert($children.length);//长度为3 ——> ul元素 span元素 p元素

for(var i=0;i<$($children[0]).children().length;i++){

alert($($children[0]).children()[i].innerHTML);

}

})

</script>

### 10）查找节点

find()方法，用来查询当前对象中的子元素节点，此方法中接收的是一个选择器参数

举例1：

<div id=”d1”>

<ul>

<li>1111</li>

<li>2222</li>

<li>3333</li>

</ul>

<span>abc</span>

<p><span>hhh</span></p>

</div>

<script type=”text/javascript”>

$(function(){

var $ul = $(“#d1”).find(“ul”);//获取当前元素对象中所有的ul对象包括子对象与后代对象

var $span = $(“#d1”).find(“span”);

alert($span.text());//结果为abchhhh，两个span对象的连接。

//如果只查找第一个span对象

alert($(“#d1”).find(“span:first”).text());

alert($(“#d1”).find(“span:eq(0)”).text());

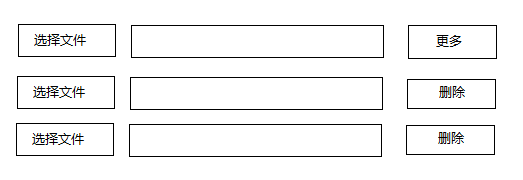
})

</script>

### 节点操作练习

要求：在页面上有个文件上传控件file对象，后面跟随一个更多按钮，点击更多按钮将在下一行创建一个新的文件上传控件file后面跟随一个删除按钮，点击删除则将此行的上传控件file对象删除掉。再点击更多按钮则将再创建一个新的文件上传控件file对象，同样后面跟随一个删除按钮。

图例：



首先使用javascript方式来实现，然后使用jquery方式来实现。

<div id=’div1’>

<input type="file" style="border:1px solid black;" /><input type="button" value="更多" onclick="add()"/>

</div>

第一种方式：

<script type=”text/javascript”>

function add(){

var div1 = document.getElementById("div1");

var input = document.createElement("input");

input.setAttribute("type","file");

input.setAttribute("style","border:1px solid black;");

var br = document.createElement("br");

var btn = document.createElement("input");

btn.setAttribute("type","button");

btn.setAttribute("value","删除");

btn.onclick=function(){

div1.removeChild(input);

div1.removeChild(btn);

div1.removeChild(br);

}

div1.appendChild(input);

div1.appendChild(btn);

div1.appendChild(br);

}

</script>

第二种方式：

<div id=’div1’>

<input type="file" style="border:1px solid black;" /><input type="button" value="更多" />

</div>

$(function(){

$(“#div input[type=button]”).click(function(){

var $input = $("<input type='file' style='border:1px solid black'>");

var $btn = $("<input type='button' value='删除'>");

var $br = $("<br>");

$btn.click(function(){

$input.remove();

$btn.remove();

$br.remove();

});

$("#div1").append($br).append($input).append($btn);

})

})

## 8、jQuery操作CSS样式

### 1）增加样式

可以使用属性attr()方法来修改元素的样式，也可以使用addClass()方法来追加元素的样式，两种方法有明显的区别

举例：

<style type=”text/css”>

.c1{

font-size:25px;

color:red;

}

.c2{

font-style:italic;

color:blue;

}

</style>

<span class=”test”>abc</span>

//=========================1

<script type=”text/javascript”>

$(function(){

$(“span”).attr(“class”,”c1”);

})

</script>

结果：<span class=”c1”>abc</span>

//==========================2

<script type=”text/javascript”>

$(function(){

$(“span”).addClass(”c1”);

})

</script>

结果：<span class=”test c1”>abc</span>

### 2）移除样式

removeClass()，不加参数表示将当前元素所使用的样式全部移除，加上参数表示将指定的样式移除。

举例：

<style type=”text/css”>

.c1{

font-size:25px;

color:red;

}

.c2{

font-style:italic;

color:blue;

}

</style>

<span class=”test c2”>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).removeClass();

})

</script>

结果：<span class=””>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”). removeClass (”c2”);

})

</script>

结果：<span class=”test”>abc</span>

### 3）切换样式

toggleClass(“参数”)，此方法用来判断当前元素是否含有指定参数的样式，如果有则将该样式移除，如果没有则将为当前元素追加此样式，从而达到切换的目的。

<span class=”test c2”>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).toggleClass(“c2”);

})

</script>

结果：<span class=”test”>abc</span>

<span class=”test”>abc</span>

<script type=”text/javascript”>

$(function(){

$(“span”).toggleClass(“c2”);

})

</script>

结果：<span class=”test c2”>abc</span>

### 4）判断是否含有样式

hasClass(“参数”)，判断当前元素是否含有指定的样式，返回一个boolean值。

举例：

<span class=”test c2”>abc</span>

<script type=”text/javascript”>

$(function(){

var flag = $(“span”).hasClass(“c2”);

})

</script>

结果：flag为 true

还可以使用is(“选择器”)方法，is(“选择器”)方法并不只用于判断是否含有样式，它是一个多使用方法，可根据选择器来判断当前元素是否处于某种状态。

例如：$(“ :checkbox:eq(1)”).is(“:checked”)，判断页面中第二个复选框是否是选中状态。

举例：<span class=”test c2”>abc</span>

<script type=”text/javascript”>

$(function(){

var flag = $(“span”).is(“.c2”);//注意不要漏掉样式前面的点，表示此样式名的选择器

})

</script>

结果：flag为 true

## 9、jQuery中的事件机制

### 1）获取焦点

Jquery提供focus()方法来进行处理，表示当前对象获得焦点。

举例：

<input type=”text” id=”username” size=”15” value=”username”/>

<input type=”password” id=”password” size=”15” />

<input type=”button” id=”bt1” onclick=”test()”/>

<script type=”text/javascript”>

$(function(){

$(“#username”).focus(function(){

var value = $(this).val();

if(value==”username”){

$(this).val(“”);

}

});

})

</script>

### 2）失去焦点

blur()，失去焦点

<input type=”text” id=”username” size=”15” />

<input type=”password” id=”password” size=”15” />

<input type=”button” id=”bt1” onclick=”test()”/>

<script type=”text/javascript”>

$(function(){

$(“#username”).blur(function(){

var value = $(this).val();

if(value==””){

$(this).val(“username”);

}

});

})

</script>

### 3）事件绑定

在不同的浏览器中，事件的处理机制有的是采用冒泡机制，有的采用的是捕获机制。

举例1：

<script type=”text/javascript”>

function bodyClick(){

alert(“body click”);

}

function divClick(){

Alert(“div click”);

}

function spanClick(){

alert(“span click”);

}

</script>

<body onclick=”bodyClick()”>

<div onclick=”divClick()”>

Span元素前的信息

<span onclick=”spanClick()”>span的信息</span>

Span元素后的信息

</div>

</body>

在jQuery中采用了事件绑定的机制，在每个元素上面进行事件的绑定，只会在该元素上面进行事件触发。

bind(“事件的类型”，处理事件的函数，参数3)方法来进行事件绑定，一般只会用到前两个参数。

举例2：

<script type=”text/javascript”>

$(function(){

$(“body”).bind(“click”,function(){

var htmlStr = $(“#d1”).html()+“<h1>body对象被点击</h1>”;

$(“#d1”).html(htmlStr);

});

$(“div:first”).bind(“click”,function(){

var htmlStr =$(“#d1”).html()+ “<h1>div对象被点击</h1>”;

$(“#d1”).html(htmlStr);

});

$(“span”).bind(“click”,function(){

var htmlStr =$(“#d1”).html()+ “<h1>span对象被点击</h1>”;

$(“#d1”).html(htmlStr);

});

})

</script>

<body>

<div>

Span元素前的信息

<span>span的信息</span>

Span元素后的信息

</div>

<div id=”d1”></div>

</body>

可以使用event提供的方法stopPropagation()方法，此方法阻止事件向外传递的行为。

举例3：

$(“body”).bind(“click”,function(event){

var htmlStr = $(“#d1”).html()+“<h1>body对象被点击</h1>”;

$(“#d1”).html(htmlStr);

});

$(“div:first”).bind(“click”,function(event){

var htmlStr =$(“#d1”).html()+ “<h1>div对象被点击</h1>”;

$(“#d1”).html(htmlStr);

event.stopPropagation();

});

$(“span”).bind(“click”,function(event){

var htmlStr =$(“#d1”).html()+ “<h1>span对象被点击</h1>”;

$(“#d1”).html(htmlStr);

event.stopPropagation();

});

举例4：

<script type=”text/javascript”>

$(function(){

$(“#h1”).bind(“click”,function(){

$(this).next().show();//show()方法可接收的参数1、slow 2、normal 3、fast 4、1000指定的毫秒

})

})

</script>

<h1 id=”h1”>点击显示隐藏的信息</h1>

<div ><h2>被隐藏的信息</h2></div>

补充：与show方法相对应的jQuery的方法中提供了一个隐藏的方法，hide()

举例5：

<script type=”text/javascript”>

$(function(){

$(“.head”).bind(“click”,function(){

var $content = $(this).next();

if($content.is(“:visible”)){

$content.hide();

}else{

$content.show();

}

})

})

</script>

<h3 class=”head”>点击标题</h3>

<div id=”div1”><h2>显示或隐藏的信息 </h2></div>

举例6：绑定鼠标悬浮与离开事件

<script type=”text/javascript”>

$(function(){

$(“.head”).bind(“mouseover”,function(){

$(this).next().show();

});

$(“.head”).bind(“mouseout”,function(){

$(this).next().hide();

});

})

</script>

<h3 class=”head”>点击标题</h3>

<div id=”div1”><h2>显示或隐藏的信息 </h2></div>

举例7：event事件中阻止默认的行为

<form action=”#”>

Username:<input type=”text” size=”10” id=”username”>

<input type=”submit” value=”提交” id=”sub”/>

</form>

<div id=”d1”></div>

<script type=”text/javascript”>

$(document).ready(function(event){

$(“#sub”).bind(“click”,function(){

var value = $(“#username”).val();

if(value==””){

$(“#d1”).html(“用户名不能为空！”);

//阻止默认事件的发生，常用return false;代替

event.preventDefault();

}

})

});

</script>

return false的实质含义有两个，一个是阻止事件的传递，一个是阻止默认的行为。

举例8：使用one()方法进行绑定，表示此事件只会执行一次，很少用

<h1 id=”h1”>点击事件</h1>

<div id=”d1”></div>

<script type=”text/javascript”>

$(document).ready(function(){

$(“#h1”).one(“click”,function(){

$(“#d1”).append(“<b>发生了点击事件</b><br>”);

}).one(“click”,function(){

$(“#d1”).append(“<b>鼠标进行了悬停</b><br>”);

}).one(“click”,function(){

$(“#d1”).append(“<b>鼠标离开了</b><br>”);

});

});

</script>

举例9：可以对一个元素同时绑定多个事件的另一种编写方式。

<h1 id=”h1” class=”test”>点击事件</h1>

<script type=”text/javascript”>

$(document).ready(function(){

$(“#h1”).bind(“mouseover mouseout”,function(){

$(this).toggleClass(“over”);

})

});

</script>

### 4）移除绑定

移除事件的绑定使用unbind()方法，方法接收一个参数，表示要移除的是哪种事件的绑定，如果没有参数表示将此对象上绑定的所有事件全部移除。

举例：

<h1 id=”h1”>点击事件</h1>

<div id=”d1”></div>

<script type=”text/javascript”>

$(document).ready(function(){

$(“#h1”).bind(“click”,function(){

$(“#d1”).html(“<b>发生了点击事件</b>”);

}).bind(“mouseover”,function(){

$(“#d1”).html(“<b>鼠标进行了悬停</b>”);

}).bind(“mouseout”,function(){

$(“#d1”).html(“<b>鼠标离开了</b>”);

});

$(“#h1”).unbind(“click”);

});

</script>

Unbind()方法接收两个参数，表示移除指定类型的指定方法名的事件函数。

举例：

<h1 id=”h1”>点击事件</h1>

<div id=”d1”></div>

<script type=”text/javascript”>

$(document).ready(function(){

$(“#h1”).bind(“click”,f1=function(){

$(“#d1”).html(“<b>发生了点击事件</b>”);

}).bind(“click”,f2=function(){

$(“#d1”).html(“<b>鼠标进行了悬停</b>”);

}).bind(“click”,f3=function(){

$(“#d1”).html(“<b>鼠标离开了</b>”);

});

$(“#h1”).unbind(“click”,f2);//表示只移除了click事件中的f2事件，f1与f3不受影响。

});

</script>

### 5）鼠标悬停与离开事件

* mouseover()

鼠标悬停事件触发的方法。

举例：

<script type=”text/javascript”>

$(function(){

$(“.head”). mouseover (function(){

$(this).next().show();

});

})

</script>

<h3 class=”head”>点击标题</h3>

<div id=”div1”><h2>显示或隐藏的信息 </h2></div>

* mouseout()

鼠标离开事件触发的方法

举例：

<script type=”text/javascript”>

$(function(){

$(“.head”). mouseout (function(){

$(this).next().hide();

});

})

</script>

<h3 class=”head”>点击标题</h3>

<div id=”div1”><h2>显示或隐藏的信息 </h2></div>

### 6）页面加载事件

jQuery中的ready()方法与js中的body元素的onload事件类似，都是等页面加载完成后触发，但是又有本质的区别，js中的onload事件是等页面结构即：dom元素对象加载完后，还有其他的图片、音频信息等二次请求信息加载完成后进行触发，而jQuery中的ready()方法是在页面结构加载完成后，即dom元素加载完成后就进行触发。

举例：

<script type=”text/javascript”>

var startTime = new Date().getTime();

function test1(){

var endTime = new Date().getTime();

var ms = endTime – startTime;

$(“#div2”).html(“<p>js window.onload事件 ，花费时间”+ms+”ms</p>”);

}

$(function(){

var endTime = new Date().getTime();

var ms = endTime – startTime;

$(“#div1”).html(“<p>jQuery ready事件 ，花费时间”+ms+”ms</p>”);

})

</script>

<body onload=”test1()”>

<img src=”” />

<img src=”” />

<div id=”div1”></div>

<div id=”div2”></div>

</body>

### 7）合成事件

1、hover(funciton 1(),function())方法，它自己完成了鼠标悬停与离开两个事件的功能，第一个参数函数实现了悬停的事件，第二参数实现了离开是事件。

举例：

<script type=”text/javascript”>

$(function(){

$(“.head”). hover (function(){

$(this).next().show();

},function(){

$(this).next().hide();

});

})

</script>

<h3 class=”head”>点击标题</h3>

<div id=”div1”><h2>显示或隐藏的信息 </h2></div>

2、toggle() 方法切换元素的可见状态。

如果被选元素可见，则隐藏这些元素，如果被选元素隐藏，则显示这些元素。

举例：

<script type=”text/javascript”>

$(function(){

$(".head").click(function(){

$(this).next().toggle();

});

});

</script>

或者

$(".head").click(function(){

$(this).next().toggle(function(){

$(this).next().show();

},function(){

$(this).next().hide();

});

});

<h3 class=”head”>点击标题</h3>

<div id=”div1”><h2>显示或隐藏的信息 </h2></div>

### 8）事件的属性

* 事件的类型

event.type

举例：

<a href=”http://www.baidu.com”>百度</a>

<script type=”text/javacript”>

$(“a”).click(function(event){

alert(event.type);//事件的类型是click

return false;

});

</script>

* 获取事件的源目标

event.target

举例：

<a href=”http://www.baidu.com”>百度</a>

<script type=”text/javacript”>

$(“a”).click(function(event){

alert(event.target.href);//事件的target源目标是a对象，它的href属性值

return false;

});

</script>

### 9）模拟操作

模拟操作：由浏览器来执行代替用户的操作，进行事件的触发。

jQuery提供的方法trigger()方法，其中接收参数为触发的事件类型。

举例：

<h1 id=”h1”>点击事件</h1>

<div id=”d1”></div>

<script type=”text/javascript”>

$(document).ready(function(){

$(“#h1”).bind(“click”,f1=function(){

$(“#d1”).html(“<b>发生了点击事件</b>”);

}).bind(“click”,f2=function(){

$(“#d1”).html(“<b>鼠标进行了悬停</b>”);

}).bind(“click”,f3=function(){

$(“#d1”).html(“<b>鼠标离开了</b>”);

});

$(“#h1”).trigger(“click”);

//当浏览器执行到此时，就触发点击事件，因由浏览器执行的，所以称浏览器模拟用户的操作 。

});

</script>

### 10）自定义事件

在JS中提供了大量的已经定义好的触发事件，比如：onclick、onmouseover、onmouserout,等等，在jQuery中同样对这些事件进行了封装，比如click、mouseover、mouserout等等。同时还提供了自定义的事件机制，就是通过trigger()方法来实现。

举例1：

<input type=”button” id=”btn” value=”test” />

<div id=”div1”></div>

<script type=”text/javascript”>

$(function(){

$(“#btn”).bind(“myClick”,function(){

$(“#div1”).html(“<h2>自定义事件触发</h2>”);

});

$(“#btn”).click(function(){

//当按钮被点击时，自定义的事件通过trigger函数被模拟操作来实现事件的触发。

$(this).trigger(“myClick”);

});

})

</script>

在自定义事件所定义的函数中，可以传递参数。

举例2：

<input type=”button” id=”btn” value=”test” />

<div id=”div1”></div>

<script type=”text/javascript”>

$(function(){

$(“#btn”).bind(“myClick”,function(event,arg1,arg2){

$(“#div1”).html(“<h2>自定义事件触发</h2>”);

});

$(“#btn”).click(function(){

//当按钮被点击时，自定义的事件通过trigger函数被模拟操作来实现事件的触发。

$(this).trigger(“myClick”,[“hello”,”world”]);

});

})

</script>

在调用方法传递参数时，jQuery自动将第一个参数event对象进行了处理，所以只需要传入后面的参数即可，传递的方式是使用[“hello”,”world”]，在对应的js中则不能省略，如：

举例3：

<input type=”button” id=”btn” value=”test” onclick=”test(event,’hello’,’world’)” />

<div id=”div1”></div>

<script type=”text/javascript”>

function test(event,arg1,arg2){

alert(arg1);

alert(arg2);

}

</script>

## 10、jQuery中的动画处理

### 1）隐藏与显示

show()方法，接收的参数：slow normal fast 接收毫秒数

hide()方法，接收的参数：slow normal fast 接收毫秒数

举例：

<h1 id="hh1" >点击事件</h1>

<div style="width:400px;height:100px;border:1px solid black;">

<h4 style="color:green">绿颜色的标题</h4>

</div>

<script type=”text/javascript”>

$("#hh1").click(function(){

var div = $(this).next();

if(div.is(":visible")){

div.hide("fast");

}else{

div.show("slow");

}

})

</script>

### 2）图层渐变

* fadeIn()方法，使得当前对象逐渐显现
* fadeOut()方法，使得当前对象逐渐消失

举例1：

<h1 id="hh1" >点击事件</h1>

<div style="width:400px;height:100px;border:1px solid black;">

<h4 style="color:green">绿颜色的标题</h4>

</div>

<script type=”text/javascript”>

$("#hh1").click(function(){

var d1 = $("#hh1").next();

if(d1.is(":visible")){

d1.fadeOut(2000);

}else{

d1.fadeIn(2000);

}

})

</script>

### 3）图层伸缩

* slideUp()方法，图层逐渐向上缩进
* slideDown()方法，图层逐渐向下拉伸

举例1：

<h1 id="hh1" >点击事件</h1>

<div style="width:400px;height:100px;border:1px solid black;">

<h4 style="color:green">绿颜色的标题</h4>

</div>

<script type=”text/javascript”>

$("#hh1").click(function(){

var d1 = $("#hh1").next();

if(d1.is(":visible")){

d1. slideUp(2000);

}else{

d1. slideDown(2000);

}

})

</script>

### 4）图层移动

* animate({css样式},间隔的毫秒数,回调函数)

图层移动调用的jQuery提供的animate()方法来实现，此方法通过样式来控制移动的方式，是向下或者向左。

为了使元素动起来，要更改元素的“left”样式属性。需要注意的是在使用animate()方法之前，为了能够影响该元素的“top”、”left”、”bottom”、”right”样式属性，必须先把元素的position样式设置为“relative”或者“absolute”。

举例1：

<div id="hh1" style="width:70px;height:70px;background:green;position:relative">点击事件</div>

$("#hh1").click(function(){

$(this).animate({left : "500px"},3000);

})

举例2：

<div id="hh1" style="width:70px;height:70px;background:green;position:relative">点击事件</div>

$("#hh1").click(function(){

$(this).animate({left : "500px"},3000)

.animate({height : “500px”},3000);

})

效果：先向右移动到距离左边500个像素，然后高度增加到500像素。

举例3：图层综合运用展现动画效果

<div id="hh1" style="width:70px;height:70px;background:green;position:relative"> </div>

$("#hh1").click(function(){

//设置图层的不透明度

$(this).css(“opacity”,”0.5”)

$(this).animate({left : "500px",height:”300px”,opacity:”1”},3000)

.animate({top : “500px”,width:”200px”},3000)

.fadeIn(3000);

})

举例4：

<div id="hh1" style="width:70px;height:70px;background:green;position:relative"> </div>

$("#hh1").click(function(){

//设置图层的不透明度

$(this).css(“opacity”,”0.5”)

$(this).animate({left : "500px",height:”300px”,opacity:”1”},3000)

.animate({top : “500px”,width:”200px”},3000,function(){

$(this).css(“border”,”5px solid blue”);

});

})

## 11、使用jQuery实现异步的调用

使用jQuery提供的$.ajax()方法来实现异步的调用，ajax()方法接收一个JSON对象的参数，里面有各个属性来表示不同的含义。

举例1：

<div id=”div1”></div>

<input type=”button” value=”test” id=”btn” />

<script type=”text/javascript”>

$(“#btn”).click(function(){

var param = {p1:’hello’,p2:’world’};

$.ajax({

type : “POST”, //请求的方式

url : “./myServlet”, //请求的资源地址

data : param,

success : function(returnData){ //成功执行后的回调函数

$(“#div1”).html(returnData);

}

});

});

</script>

在请求的servlet中将传入的参数获取，并将两个参数拼接后进行返回。

String p1 = request.getParameter(“p1”);

String p2 = request.getParameter(“p2”);

out.print(“p1:”+p1+”===p2:”+p2);

结果：

<div>p1:hello===p2:world</div>

举例2：调用jQuery的post或者get方法进行请求

<div id=”div1”></div>

<input type=”button” value=”test” id=”btn” />

$(function(){

$("#btn").click(function(){

var params = {p1:'hello',p2:'world'};

$.post("./myServlet",params,function(returnData,status){

$("#div1").html(returnData);

});

});

});

举例3：使用jQuery处理返回格式是XML类型的数据

$(function(){

$("#btn").click(function(){

var param = new Object();

param.name = "zhangsan";

$.post("./myServlet",param,function(returnData,status){

var owner = $(returnData).find("owner").text();

var orderTime = $(returnData).find("dataTime").text();

var totalPrice = $(returnData).find("totalPrice").text();

var $ta = "<table border='1'><tr><th>订单人</th><th>订单时间</th><th>订单价格</th></tr>";

$ta = $ta + "<tr><td>"+owner+"</td><td>"+orderTime+"</td><td>"+totalPrice+"</td></tr></table>";

$("#div1 :first-child").remove();

$("#div1").append($ta);

});

});

});着火了

请求后台的Servlet对象

private void process(HttpServletRequest req, HttpServletResponse resp){

try {

String name = req.getParameter("name");

Order order = new Order();

if("zhangsan".equals(name)){

order.setOwner("张三");

order.setDataTime("2014-03-01");

order.setTotalPrice(1237.50);

}else{

order.setOwner("李四");

order.setDataTime("2014-03-02");

order.setTotalPrice(2334.75);

}

Document doc = new Document();

Element root = new Element("order");

doc.setRootElement(root);

Element owner = new Element("owner");

owner.setText(order.getOwner());

root.addContent(owner);

Element dataTime = new Element("dataTime");

dataTime.setText(order.getDataTime());

root.addContent(dataTime);

Element totalPrice = new Element("totalPrice");

totalPrice.setText(String.valueOf(order.getTotalPrice()));

root.addContent(totalPrice);

XMLOutputter xmlOut = new XMLOutputter();

xmlOut.setFormat(Format.getPrettyFormat());

resp.setHeader("pragma","no-cache");

resp.setHeader("cache-control", "no-cache");

resp.setContentType("text/xml");

PrintWriter out = resp.getWriter();

xmlOut.output(doc, out);

out.flush();

out.close();

} catch (Exception e) {

e.printStackTrace();

}

}

调用的javabean类

final class Order{

private String dataTime;

private String owner;

private double totalPrice;

public String getDataTime() {

return dataTime;

}

public void setDataTime(String dataTime) {

this.dataTime = dataTime;

}

public String getOwner() {

return owner;

}

public void setOwner(String owner) {

this.owner = owner;

}

public double getTotalPrice() {

return totalPrice;

}

public void setTotalPrice(double totalPrice) {

this.totalPrice = totalPrice;

}

}

举例4：使用jQuery处理复杂格式的xml数据

<script type="text/javascript" src="./jQuery/jquery-2.0.3.js" ></script>

<script type="text/javascript">

$(function(){

$("#btn").click(function(){

var param = new Object();

param.name = "zhangsan";

$.post("./myServlet",param,function(returnData,status){

$("#div1").children().remove();

var owner = $(returnData).children().attr("owner");

var dataTime = $(returnData).children().attr("dataTime");

var orderCode = $(returnData).children().attr("orderCode");

var totalPrice = $(returnData).children().attr("totalPrice");

var div = "<div>用户名："+owner+"&nbsp;&nbsp;&nbsp;&nbsp;订单日期："+dataTime;

div += "&nbsp;&nbsp;&nbsp;&nbsp;订单编号："+orderCode+"&nbsp;&nbsp;&nbsp;&nbsp;订单总价："+totalPrice;

$("#div1").append(div);

var tableData = "<table border='1px' width='600px'><tr><th width='80px'>商品编号</th><th width='100px'>商品名称</th>";

tableData += "<th width='100px'>商品进价</th><th width='100px'>商品单价</th><th width='100px'>商品数量</th></tr>";

var itemList = $(returnData).find("item ");

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

var itemCode = $(itemList[i]).attr("itemCode");

var itemName = $(itemList[i]).attr("itemName");

var nums = $(itemList[i]).attr("nums");

var purchPrice = $(itemList[i]).attr("purchPrice");

var singlePrice = $(itemList[i]).attr("singlePrice");

tableData += "<tr><td>"+itemCode+"</td><td>"+itemName+"</td><td>"+purchPrice+"</td>";

tableData += "<td>"+singlePrice+"</td><td>"+nums+"</td></tr>";

}

tableData += "</table>";

$("#div1").append(tableData);

});

});

});

</script>

文件2：请求的后台的servlet

private void process(HttpServletRequest req, HttpServletResponse resp){

try {

String name = req.getParameter("name");

Order order = new Order();

List<Item> itemList = new ArrayList<Item>();

if("zhangsan".equals(name)){

order.setOrderCode("DJ000431");

order.setDataTime("2014-02-22");

order.setOwner("张三");

order.setTotalPrice(1234.50);

Item it1 = new Item();

it1.setItemCode("SP001");

it1.setItemName("鼠标垫");

it1.setNums("3");

it1.setPurchPrice("2.50");

it1.setSinglePrice("4.00");

itemList.add(it1);

Item it2 = new Item();

it2.setItemCode("SP002");

it2.setItemName("键盘");

it2.setNums("6");

it2.setPurchPrice("12.50");

it2.setSinglePrice("24.00");

itemList.add(it2);

Item it3 = new Item();

it3.setItemCode("SP003");

it3.setItemName("显示器");

it3.setNums("12");

it3.setPurchPrice("670.00");

it3.setSinglePrice("884.00");

itemList.add(it3);

order.setItemList(itemList);

}

Document doc = new Document();

Element root = new Element("order");

doc.setRootElement(root);

root.setAttribute("owner", order.getOwner()).setAttribute("dataTime", order.getDataTime())

.setAttribute("orderCode",order.getOrderCode())

.setAttribute("totalPrice",String.valueOf(order.getTotalPrice()));

for(int i=0;i<itemList.size();i++){

Element item = new Element("item");

item.setAttribute("itemCode",itemList.get(i).getItemCode())

.setAttribute("itemName",itemList.get(i).getItemName()) .setAttribute("nums",itemList.get(i).getNums())

.setAttribute("purchPrice",itemList.get(i).getPurchPrice())

.setAttribute("singlePrice",itemList.get(i).getSinglePrice());

root.addContent(item);

}

XMLOutputter xmlOut = new XMLOutputter();

xmlOut.setFormat(Format.getPrettyFormat());

resp.setHeader("pragma","no-cache");

resp.setHeader("cache-control", "no-cache");

resp.setContentType("text/xml");

PrintWriter out = resp.getWriter();

xmlOut.output(doc, out);

out.flush();

out.close();

} catch (Exception e) {

e.printStackTrace();

}

}

文件3：javabean文件

final class Order{

private String dataTime;

private String owner;

private String orderCode;

private double totalPrice;

private List<Item> itemList;

public String getOrderCode() {

return orderCode;

}

public void setOrderCode(String orderCode) {

this.orderCode = orderCode;

}

public List<Item> getItemList() {

return itemList;

}

public void setItemList(List<Item> itemList) {

this.itemList = itemList;

}

public String getDataTime() {

return dataTime;

}

public void setDataTime(String dataTime) {

this.dataTime = dataTime;

}

public String getOwner() {

return owner;

}

public void setOwner(String owner) {

this.owner = owner;

}

public double getTotalPrice() {

return totalPrice;

}

public void setTotalPrice(double totalPrice) {

this.totalPrice = totalPrice;

}

}

final class Item{

private String itemCode;

private String itemName;

private String singlePrice;

private String nums;

private String purchPrice;

public String getItemCode() {

return itemCode;

}

public void setItemCode(String itemCode) {

this.itemCode = itemCode;

}

public String getItemName() {

return itemName;

}

public void setItemName(String itemName) {

this.itemName = itemName;

}

public String getSinglePrice() {

return singlePrice;

}

public void setSinglePrice(String singlePrice) {

this.singlePrice = singlePrice;

}

public String getNums() {

return nums;

}

public void setNums(String nums) {

this.nums = nums;

}

public String getPurchPrice() {

return purchPrice;

}

public void setPurchPrice(String purchPrice) {

this.purchPrice = purchPrice;

}

}

# 三、JSON

## 1、简介

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。JSON 使用 Javascript语法来描述数据对象，但是 JSON 仍然独立于语言和平台。JSON 解析器和 JSON 库支持许多不同的编程语言。 目前非常多的动态（PHP，JSP，.NET）编程语言都支持JSON。

举例1:

<script type=”text/javascript”>

var JSONObject = {

"name":"John Johnson",

"street":"Oslo West 555",

"age":33,

"phone":"555 1234567"

};

document.getElementById("jname").innerHTML=JSONObject.name

document.getElementById("jage").innerHTML=JSONObject.age

document.getElementById("jstreet").innerHTML=JSONObject.street

document.getElementById("jphone").innerHTML=JSONObject.phone

</script>

<p id=”jname”></p>

<p id=”jage”></p>

<p id=”jstreet”></p>

<p id=”jphone”></p>

### 1）JSON与 XML的差异

相同之处

* + JSON 是纯文本
  + JSON 具有"自我描述性"（人类可读）
  + JSON 具有层级结构（值中存在值）
  + JSON 可通过 JavaScript 进行解析
  + JSON 数据可使用 AJAX 进行传输

不同之处

* 没有结束标签
* 更短
* 读写的速度更快
* 能够使用内建的 JavaScript JSON.parse方法进行解析
* 使用数组
* 不使用保留字

### 2）为什么使用 JSON？

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用：

使用 XML

读取 XML 文档

使用 XML DOM 来循环遍历文档

读取值并存储在变量中

使用 JSON

读取 JSON 字符串

用 eval() 处理 JSON 字符串

使用JSON完全可以代替XML对象

举例1:

<person>

<name>zhangsan</name>

<age>21</age>

<gender>male</gender>

</person>

jsonObj = {

name : ’zhangsan’,

age : 21,

gender : ’male’

}

举例2：

<persons>

<person>

<name>zhangsan</name>

<age>21</age>

<gender>male</gender>

</person>

<person>

<name>zhangsan</name>

<age>21</age>

<gender>male</gender>

</person>

</persons>

jsonObj =[

{

name : ’zhangsan’,

age : 21,

gender : ’male’

},

{

name : ’zhangsan’,

age : 21,

gender : ’male’

}

]

举例3：

<person>

<name>zhangsan</name>

<age>21</age>

<gender>male</gender>

<results>

<chinese>86</chinese>

<math>88</math>

<english>83</english>

<results>

</person>

jsonObj = {

name : ‘zhangsan’,

age : 21,

gender : ‘male’,

result : {

chinese : 86,

math : 88,

english : 83

}

}

## 2、JSON的语法

JSON 语法是 JavaScript 语法的子集。

### 1）JSON的语法规则

JSON 语法是 JavaScript 对象表示法语法的子集。

* + 数据在名称/值对中
  + 数据由逗号分隔
  + 花括号保存对象
  + 方括号保存数组

### 2）JSON 名称/值对

JSON 数据的书写格式是：名称/值对。

名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值：

"firstName" : "John"

这很容易理解，等价于这条 JavaScript 语句：

firstName = "John"

### 3）JSON的值

* 数字（整数或浮点数）
* 字符串（在双引号中）
* 逻辑值（true 或 false）
* 数组（在方括号中）
* 对象（在花括号中）
* null

### 4）JSON 对象

JSON 对象在花括号中书写：

对象可以包含多个名称/值对：

{ "firstName":"John" , "lastName":"Doe" }

### 5）JSON数组

JSON 数组在方括号中书写：

数组可包含多个对象：

{

"employees": [{ "firstName":"John" , "lastName":"Doe" },

{ "firstName":"Anna" , "lastName":"Smith" },

{ "firstName":"Peter" , "lastName":"Jones" }]

}

### 6）JSON 使用 JavaScript 语法

因为 JSON 使用 JavaScript 语法，所以无需额外的软件就能处理 JavaScript 中的 JSON。通过 JavaScript，您可以创建一个对象数组，并像这样进行赋值：

举例1：

var employees = [

{ "firstName":"John" , "lastName":"Doe" },

{ "firstName":"Anna" , "lastName":"Smith" },

{ "firstName":"Peter" , "lastName": "Jones" }

];

可以像这样访问 JavaScript 对象数组中的第一项：

employees[0].lastName;

返回的内容是：:

Doe

可以像这样修改数据：

employees[0].firstName = "Jonatan";

## 3、JSON 文件

JSON 文件的文件类型是 ".json"

JSON 文本的 MIME 类型是 "application/json"

## 4、JSON文本转换为JS对象

将JSON文本（JSON字符串）转换为JS对象，使用JS提供的eval()方法。eval() 函数使用的是 JavaScript 编译器，可解析 JSON 文本，然后生成 JavaScript 对象。必须把文本包围在括号中，这样才能避免语法错误。

举例1：

var text = "{'username':'zhangsan','age':21,'gender':'male'}";

var jsonObj = eval("("+text+")");

alert(jsonObj.username);

## 5、JSON的使用

Java中使用JSON，使用第三方的类库

一、org.json库

首先将第三方的java类引入进来，在JSON官方网站[www.json.org中提供了java对json](http://www.json.org中提供了java对json)对象处理的源文件，将其下载回来引入到应用中。

JSONObject类，专门用来处理JSON对象

举例1：

public static void main(String[] args) {

String json = "{username:'zhangsan',age:20,gender:'male'}";

JSONObject obj = new JSONObject(json);

String username = obj.getString("username");

String gender = obj.getString("gender");

int age = obj.getInt("age");

System.out.println("username:"+username);

System.out.println("gender:"+gender);

System.out.println("age:"+age);

}

JSONArray类，专门用来处理JSON数组

举例2：

public static void main(String[] args) {

String json = "[{username:'zhangsan',age:20},{username:'lisi',age:25}]";

JSONArray arr = new JSONArray(json);

for(int i=0;i<arr.length();i++){

JSONObject jsonObj = arr.optJSONObject(i);

String name = jsonObj.getString("username");

int age = jsonObj.getInt("age");

System.out.println("name:"+name);

System.out.println("age"+age);

}

}

举例3：

将Java对象转换成json字符串

**package** com.langsin.test;

**import** org.json.JSONObject;

**public** **class** JsonTest {

**public** **static** **void** main(String[] args) {

Student st = **new** Student();

st.getList().add(**new** Student());

st.getList().add(**new** Student());

st.getList().add(**new** Student());

JSONObject json = **new** JSONObject(st);

System.*out*.println(json.toString());

}

}

二、google-gson类库

可以将JSON对象转换成一个java对象，也可以将一个java对象转换成JSON字符串。

将一个java对象转换为一个json对象使用toJson()方法。

举例1：

public static void main(String[] args) {

Student st = new Student();

st.getList().add("你好");

st.getList().add("我好");

st.getList().add("大家好");

Gson gson = new Gson();

String jsonStr = gson.toJson(st);

System.out.println(jsonStr);

}

输出结果：{"classNo":990301,"name":"Jhon","age":21,"list":["你好","我好","大家好"]}

Student类：

package com.langsin.test;

import java.util.ArrayList;

import java.util.List;

public class Student {

public Student(){

this.classNo = 990301;

this.name = "Jhon";

this.age = 21;

}

private long classNo;

private String name;

private int age;

private List<String> list = new ArrayList<String>();

public List<String> getList() {

return list;

}

public void setList(List<String> list) {

this.list = list;

}

public long getClassNo() {

return classNo;

}

public void setClassNo(long classNo) {

this.classNo = classNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

举例2：将一个JSON串转换成JavaBean对象

**package** com.langsin.test;

**import** com.google.gson.Gson;

**public** **class** JsonTest {

**public** **static** **void** main(String[] args) {

String str = "{username:'zhangsan',age:'24',list:['123','abc','ddd']}";

Gson gson = **new** Gson();

Student st = gson.fromJson(str, Student.**class**);

System.*out*.println(st.getAge());

System.*out*.println(st.getName());

}

}

三、使用jQuery提供的AJAX方法来获取服务器端的JSON对象

举例1：请求的后台的servlet文件

private void process(HttpServletRequest req, HttpServletResponse resp){

try {

String name = req.getParameter("name");

Order order = new Order();

List<Item> itemList = new ArrayList<Item>();

if("zhangsan".equals(name)){

order.setOrderCode("DJ000431");

order.setDataTime("2014-02-22");

order.setOwner("张三");

order.setTotalPrice(1234.50);

Item it1 = new Item();

it1.setItemCode("SP001");

it1.setItemName("鼠标垫");

it1.setNums("3");

it1.setPurchPrice("2.50");

it1.setSinglePrice("4.00");

itemList.add(it1);

Item it2 = new Item();

it2.setItemCode("SP002");

it2.setItemName("键盘");

it2.setNums("6");

it2.setPurchPrice("12.50");

it2.setSinglePrice("24.00");

itemList.add(it2);

Item it3 = new Item();

it3.setItemCode("SP003");

it3.setItemName("显示器");

it3.setNums("12");

it3.setPurchPrice("670.00");

it3.setSinglePrice("884.00");

itemList.add(it3);

order.setItemList(itemList);

}

Gson gson = new Gson();

String json = gson.toJson(order);

resp.setHeader("pragma","no-cache");

resp.setHeader("cache-control", "no-cache");

resp.setContentType("application/json");

PrintWriter out = resp.getWriter();

out.print(json);

out.flush();

out.close();

} catch (Exception e) {

e.printStackTrace();

}

}

final class Order{

private String dataTime;

private String owner;

private String orderCode;

private double totalPrice;

private List<Item> itemList;

public String getOrderCode() {

return orderCode;

}

public void setOrderCode(String orderCode) {

this.orderCode = orderCode;

}

public List<Item> getItemList() {

return itemList;

}

public void setItemList(List<Item> itemList) {

this.itemList = itemList;

}

public String getDataTime() {

return dataTime;

}

public void setDataTime(String dataTime) {

this.dataTime = dataTime;

}

public String getOwner() {

return owner;

}

public void setOwner(String owner) {

this.owner = owner;

}

public double getTotalPrice() {

return totalPrice;

}

public void setTotalPrice(double totalPrice) {

this.totalPrice = totalPrice;

}

}

final class Item{

private String itemCode;

private String itemName;

private String singlePrice;

private String nums;

private String purchPrice;

public String getItemCode() {

return itemCode;

}

public void setItemCode(String itemCode) {

this.itemCode = itemCode;

}

public String getItemName() {

return itemName;

}

public void setItemName(String itemName) {

this.itemName = itemName;

}

public String getSinglePrice() {

return singlePrice;

}

public void setSinglePrice(String singlePrice) {

this.singlePrice = singlePrice;

}

public String getNums() {

return nums;

}

public void setNums(String nums) {

this.nums = nums;

}

public String getPurchPrice() {

return purchPrice;

}

public void setPurchPrice(String purchPrice) {

this.purchPrice = purchPrice;

}

}

文件2：前台处理返回的json字符串

$(function(){

$("#btn").click(function(){

var param = new Object();

param.name = "zhangsan";

$.post("./myServlet",param,function(returnData,status){

$("#div1").children().remove();

var owner = returnData.owner;

var dataTime = returnData.dataTime;

var orderCode = returnData.orderCode;

var totalPrice = returnData.totalPrice;

var div = "<div>用户名："+owner+"&nbsp;&nbsp;&nbsp;&nbsp;订单日期："+dataTime;

div += "&nbsp;&nbsp;&nbsp;&nbsp;订单编号："+orderCode+"&nbsp;&nbsp;&nbsp;&nbsp;订单总价："+totalPrice;

$("#div1").append(div);

var tableData = "<table border='1px' width='600px'><tr><th width='80px'>商品编号</th><th width='100px'>商品名称</th>";

tableData += "<th width='100px'>商品进价</th><th width='100px'>商品单价</th><th width='100px'>商品数量</th></tr>";

var itemList = returnData.itemList;

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

var itemCode = itemList[i].itemCode;

var itemName = itemList[i].itemName;

var nums = itemList[i].nums;

var purchPrice = itemList[i].purchPrice;

var singlePrice = itemList[i].singlePrice;

tableData += "<tr><td>"+itemCode+"</td><td>"+itemName+"</td><td>"+purchPrice+"</td>";

tableData += "<td>"+singlePrice+"</td><td>"+nums+"</td></tr>";

}

tableData += "</table>";

$("#div1").append(tableData);

});

});

});

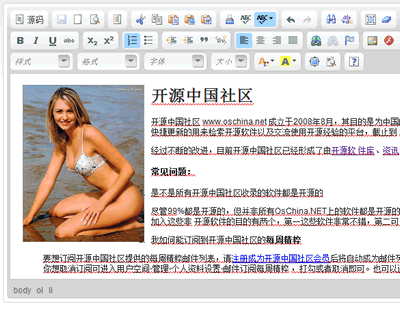
# 四、HTML可视化编辑器

## 1、CKEditor

可视化HTML编辑器 CKEditor。CKEditor是新一代的FCKeditor，是一个重新开发的版本。CKEditor是全球最优秀的网页在线文字编辑器之一，因其惊人的性能与可扩展性而广泛的被运用于各大网站。

可配合使用的扩展有 [文件管理器KCFinder](http://www.oschina.net/p/kcfinder)。

效果如下：



参考：<http://www.oschina.net/p/ckeditor> 开源中国社区

## 2、kindEditor

KindEditor是一套开源的HTML可视化编辑器，主要用于让用户在网站上获得所见即所得编辑效果，兼容IE、 Firefox、Chrome、 Safari、Opera等主流浏览器。 KindEditor使用JavaScript编写，可以无缝的与Java、.NET、PHP、ASP等程序接合。

KindEditor 4.1.5 发布，在线富文本编辑器

KindEditor 是一套开源的在线HTML编辑器，主要用于让用户在网站上获得所见即所得编辑效果，开发人员可以用 KindEditor 把传统的多行文本输入框(textarea)替

