# Web前端开发

主流应用框架：Extjs/JQuery

JQuery: 免费开源，视图与代码分离：轻量级的框架。

官方提供JQuery.js核心库和JQuery-UI.js以及移动开发的支持库。

如果要使用JQuery制作系统的UI的时候还需自定义组件或者使用各种第三方开源插件（组件）。

Extjs：半开源，视图与代码完全分离：重量级的框架。

官方提供基本可以满足所有的开发需求的核心库以及组件库。还提供了几百个例子项目。

# **jQuery**

**jQuery 是一个 JavaScript 函数库。**

**jQuery 极大地简化了 JavaScript 编程。**

jQuery 库包含以下特性：

* HTML 元素选取
* HTML 元素操作
* CSS 操作
* HTML 事件函数
* JavaScript 特效和动画
* HTML DOM 遍历和修改
* AJAX
* Utilities

**jQuery 很容易学习。学习前必须掌握**：ＨＴＭＬ、ＣＳＳ、ＪＳ

Demo01:

1. 引入核心函数库JQuery-1.xx.x.js
2. 总是在就绪函数中编程。$(document).ready();
3. 当document就绪后执行function： $(document).ready(function(){ });
4. 学习各种例子，只需要关注代码和视图两大部分。

## jQuery 语法

jQuery 语法是为 HTML 元素的选取编制，可以对元素执行某些操作。

基础语法是：$(selector).action()

* 美元符号定义 jQuery
* 选择符（selector）“查询”和“查找” HTML 元素
* jQuery action() 执行对元素的操作（事件绑定，动画效果，插件函数）

### 实例

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$("p.test").hide() - 隐藏所有 class="test" 的段落

$("#test").hide() - 隐藏所有 id="test" 的元素

## 文档就绪函数

您也许已经注意到在我们的实例中的所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){

--- jQuery functions go here ----

});

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

下面的例子演示了在文档完全加载之前运行函数的话，操作失败的情况：

* 试图隐藏一个不存在的元素。
* 获得未完全加载的图像的大小。

# jQuery 事件函数

jQuery 事件处理函数是 jQuery 中的核心函数。

事件处理函数是当 HTML 中发生事件时自动被调用的函数。由“事件”（event）“触发”（triggered）是经常被用到的术语。

只需要记忆常用的事件。

点击，失去焦点，选择，鼠标悬停，表单提交…

# jQuery 效果

隐藏、显示、切换

$(selector).hide(speed,callback)

$(selector).show(speed,callback)

$(selector).toggle(speed,callback)

上下滑动、切换

$(selector).slideDown(speed,callback)

$(selector).slideUp(speed,callback)

$(selector).slideToggle(speed,callback)

渐入淡出

$(selector).fadeIn(speed,callback)

$(selector).fadeOut(speed,callback)

$(selector).fadeTo(speed,opacity,callback)

动画

$(selector).animate({params},[duration],[easing],[callback])

speed参数可以设置这些值："slow", "fast", "normal" 或 毫秒。

callback 参数是在当前函数完成之后被执行的函数名称，称为回调函数，确保函数执行的顺序。

# jQuery 插件

1. 主流浏览器的支持
2. 版本兼容性，很多插件由第三方的团队开发，不一定兼容最新版本的JQuery。

# JQuery官方提供的插件库：

jQuery UI ：<http://jqueryui.com/>

jQuery Mobile ：<http://jquerymobile.com/>

jQuery Form ：<http://plugins.jquery.com/form/>

jQuery Validate：<http://plugins.jquery.com/validate/>

# JQuery主流的第三方插件库：

jQuery Easy UI：<http://www.jeasyui.com/>

jQuery layout：<http://layout.jquery-dev.com/>

jQuery jqGrid：<http://www.trirand.com/blog/>

jQuery jqGrid Demo：<http://www.trirand.com/blog/jqgrid/jqgrid.html>

jQuery jsTree：<http://www.jstree.com/>

OPOA

One Page, One Application缩写为OPOA  
含义很简单：一个页面就是一个应用。在众多的基于Web的MIS系统中，没有人关心页面的组织形式；大多数稍微复杂的MIS系统，都采用分祯 (Frame)的方式来组织页面，这样，在进行业务操作的时候，url的变化表现在一个框架页面内，从浏览器的地址看起来，只有一个地址；更有甚者，一些应用干脆弹出一个去掉了浏览器菜单、工具条、地址栏、状态栏的窗口（比如招商银行、民生银行的网上银行系统），连地址都看不见。因此，一个页面就是一个应用，从用户的角度来说，对于操作型系统，是一种非常自然的体现。

１．导入样式和类库：注意版本的兼容性！

<!--CSS-->

<link href="css/jquery-ui.css" rel="stylesheet">

<style >

body{

font: 60.5% "Trebuchet MS", sans-serif;

}

</style>

<!--JS-->

<!--1.核心库-->

<script type="text/javascript" src="js/jquery-1.11.2.js"></script>

<!--2.插件库-->

<!--layout插件1.4-->

<script type="text/javascript" src="js/jquery.layout-latest.js"></script>

<!--ui插件1.11.2-->

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

### Demo01:添加布局和菜单栏

JS:

$(document).ready(function(){

$('body').layout({ applyDefaultStyles: true });

$( "#accordion" ).accordion();

});

HTML:

<div class="ui-layout-center">Center</div>

<div class="ui-layout-north">North</div>

<div class="ui-layout-south">South</div>

<div class="ui-layout-east">East</div>

<!--菜单栏　west -->

<div class="ui-layout-west">

<div id="accordion">

<h3>订单管理</h3>

<div></div>

<h3>财务管理</h3>

<div></div>

<h3>系统管理</h3>

<div></div>

</div>

</div>

### Demo02:添加菜单栏的选项

\jquery-ui-1.9.0.custom\development-bundle\demos\selectable

复制样式表：为了避免ID冲突，使用Class选择器

.selectable .ui-selecting { background: #FECA40; }

.selectable .ui-selected { background: #F39814; color: white; }

.selectable { list-style-type: none; margin: 0; padding: 0; width: 60%; }

.selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; height: 18px; }

JS: 使用Class选择器

$( ".selectable" ).selectable({

stop: function() {

$( ".ui-selected", this ).each(function() {

//alert(1);

});

}

});

HTML:

<!--菜单栏-->

<div class="ui-layout-west">

<div id="accordion">

<h3>订单管理</h3>

<div>

<ol class="selectable">

<li class="ui-widget-content">创建订单</li>

<li class="ui-widget-content">产品管理</li>

</ol>

</div>

<h3>财务管理</h3>

<div>

<ol class ="selectable">

<li class="ui-widget-content">应付账款</li>

<li class="ui-widget-content">应收账款</li>

</ol>

</div>

<h3>系统管理</h3>

<div>

<ol class ="selectable">

<li class="ui-widget-content">用户管理</li>

<li class="ui-widget-content">授权管理</li>

</ol>

</div>

</div>

</div>

### Demo03:升级菜单栏的点击事件

目的是点击菜单时的选项时可以获得每个模块唯一的ID.通过ID控制Center层内对应模块的面板的显示。

JS:

$( ".selectable" ).selectable({

stop: function() {

$( ".ui-selected", this ).each(function(index, domEle) {

alert( $(domEle).attr('id'));

});

}

});

HTML:

<!--菜单栏-->

<div class="ui-layout-west">

<div id="accordion">

<h3>订单管理</h3>

<div>

<ol class="selectable">

<li class="ui-widget-content" id="order">创建订单</li>

<li class="ui-widget-content" id="product">产品管理</li>

</ol>

</div>

<h3>财务管理</h3>

<div>

<ol class="selectable">

<li class="ui-widget-content" id="payable">应付账款</li>

<li class="ui-widget-content" id="receivable">应收账款</li>

</ol>

</div>

<h3>系统管理</h3>

<div>

<ol class="selectable">

<li class="ui-widget-content" id="user">用户管理</li>

<li class="ui-widget-content" id="authorize">授权管理</li>

</ol>

</div>

</div>

</div>

### Demo04:通过菜单栏的点击事件控制中间面板的显示

1. 为每个模块添加唯一中间模块层，并且为每个层添加唯一标识id+”Panel”

例如：

<ol class="selectable">

<li class="ui-widget-content" id="payable">应付账款</li> payablePanel

<li class="ui-widget-content" id="receivable">应收账款</li> receivablePanel

</ol>

HTML: class为了统一控制全部隐藏。Id为精确控制唯一div显示。

<div class="ui-layout-center">

<div class="panel" id="orderPanel">订单模块：Grid\Tree\Form\Chart</div>

<div class="panel" id="productPanel">产品模块</div>

<div class="panel" id="payablePanel">应付账款模块</div>

<div class="panel" id="receivablePanel">应收账款模块</div>

<div class="panel" id="userPanel">用户模块</div>

<div class="panel" id="authorizePanel">授权模块</div>

</div>

JS:

隐藏全部：

$(".panel").hide();//加载的时候只执行一次

$( ".selectable" ).selectable({

stop: function() {

$( ".ui-selected", this ).each(function(index, domEle) {

//alert( $(domEle).attr('id'));

$(".panel").hide();//每次隐藏前需要隐藏全部再显示

var id = $(domEle).attr('id');

$("#"+id+"Panel").show();//显示对应的模块面板

});

}

});

### Demo05:为模块面板添加Tabs

**JS**

$( ".tabs" ).tabs();

HTML:例子

<div class="panel" id="userPanel">用户模块</div>

为上方的中间层添加tabs层：

<div class="panel" id="userPanel">

<div class="tabs">

<ul>

<li><a href="#userPanelTabs-1">用户模块</a></li>

</ul>

<div id="userPanelTabs-1">

Grid\Tree\Form\Chart\tabs\dialog

</div>

</div>

</div>

### Demo06:为模块面板添加Grid

**添加依赖库：**

<!--jqGrid CSS-->

<link href="css/ui.jqgrid.css" rel="stylesheet">

<!--核心库及UI库的后面添加-->

<!--jqGrid插件国际化文件-->

<script type="text/javascript" src="js/i18n/grid.locale-cn.js"></script>

<!--jqGrid插件4.7.0-->

<script type="text/javascript" src="js/jquery.jqGrid.src.js"></script>

HTML：

<div class="panel" id="userPanel">

<div class="tabs">

<ul>

<li><a href="#userPanelTabs-1">用户模块</a></li>

</ul>

<div id="userPanelTabs-1">

<table id="userPanelGrid"></table>

<div id="userPanelGridPager"></div>

</div>

</div>

</div>

JS:

$("#userPanelGrid").jqGrid({

datatype: "local",

//JSON数据格式

// url:'findAll.action', datatype: "json",

height: 250,

colNames:['Inv No','User Name', 'Password'],

colModel:[

{name:'id',index:'id', width:60, sorttype:"int"},

{name:'userName',index:'userName', width:100},

{name:'password',index:'password', width:100}

],

pager: '#userPanelGridPager',

rowNum:50,

rowList:[10,20,30,40,50],

viewrecords: true,

multiselect: true,

sortname: 'id',

sortorder: "desc",

caption: "User Panel Grid"

});

### Demo07:为Grid设置自适应宽度

JS：

//window resize后组件宽度自适应

$(window).resize(function(){

$(window).unbind("onresize");

//window resize后重新设置所有grid的宽度

$(".panelGrid").setGridWidth($(window).width() - 265);

//其他插件/组件的宽度设置

$(window).bind("onresize", this);

});

每个Grid初始化的时候自动宽度，列的自适应

$("#userPanelGrid").jqGrid({

//…

height: 250,

width:window.screen.availWidth-265,

autowidth: false,

shrinkToFit: true,

//…

HTML：

<table id="userPanelGrid" class="panelGrid"></table>

<div id="userPanelGridPager"></div>

### Demo08:OPOA重构

为了以后系统的维护和扩展，需要对系统进行代码重构：

1.代码分离，一个插件一个JS文件。并且按照模块名/插件类型名的方式保存。

例如：system/user/grid/ 意思为用户模块的grid插件存放的文件夹。

2.命名规范：因为每个插件的id是唯一的，所以文件名也最好按照id名来定义。

例如：$("#userPanelGrid").jqGrid(); 保存到userPanelGrid.js的文件。

<!--3.系统UI核心代码-->

<script type="text/javascript" src="js/system/core/system\_core.js"></script>

<!--4.系统每个模块的入口程序-->

<script type="text/javascript" src="js/system/user/grid/userPanelGrid.js"></script>

### system\_core.js

$(document).ready(function(){

//window resize后组件宽度自适应

$(window).resize(function(){

$(window).unbind("onresize");

//window resize后重新设置所有grid的宽度

$(".panelGrid").setGridWidth($(window).width() - 265);

$(window).bind("onresize", this);

});

$('body').layout({ applyDefaultStyles: true });

$( "#accordion" ).accordion();

$( ".tabs" ).tabs();

$(".panel").hide();

$( ".selectable" ).selectable({

stop: function() {

$( ".ui-selected", this ).each(function(index, domEle) {

//alert( $(domEle).attr('id'));

$(".panel").hide();

var id = $(domEle).attr('id');

$("#"+id+"Panel").show();

});

}

});

});

### userPanelGrid.js

$(document).ready(function(){

$("#userPanelGrid").jqGrid({

datatype: "local",

//JSON数据格式

// url:'findAll.action', datatype: "json",

height: 250,

//autoheight: true,

width:window.screen.availWidth-265,

autowidth: false,

shrinkToFit: true,

colNames:['Inv No','User Name', 'Password'],

colModel:[

{name:'id',index:'id', width:60, sorttype:"int"},

{name:'userName',index:'userName', width:100},

{name:'password',index:'password', width:100}

],

pager: '#userPanelGridPager',

rowNum:50,

rowList:[10,20,30,40,50],

viewrecords: true,

multiselect: true,

sortname: 'id',

sortorder: "desc",

hidegrid: false,

caption: "User Panel Grid"

});

});

### Demo09:OPOA性能优化

1. 以后发布的文件最好使用压缩后的js/css文件。
2. 添加命名空间管理。
3. 加入动态加载js的功能，类似于Java包的导入。避免一次过加载太多的js文件。
4. 按需初始化及渲染插件。
5. 多浏览器及分辨率测试。
   1. 13寸以上或者
   2. 800\*600以上
   3. 各种主流浏览器。http://static.jstree.com/3.0.8/assets/images/browsers.pngAll modern browsers are supported, as well as IE8.
   4. **注意多一个符号，少一个符号导致的问题。**
   5. **Html5 支持 <!DOCTYPE HTML>**

### 按需初始化及渲染插件：

写法升级：替换为第二个例子的js文件：

<!--3.系统UI核心代码-->

<script type="text/javascript" src="js/system/core/system\_core2.js"></script>

<!--4.系统每个模块的入口程序-->

<script type="text/javascript" src="js/system/user/grid/userPanelGrid2.js"></script>

1.把插件写入到对象：userPanelGrid2.js 下面为模板写法：

userPanelGrid2= function(){

return {

init : function(){

//grid

alert("init");

}

};

};

**这样写的好处在于js文件加载的时候并不会立即执行**grid**的初始化和渲染。**

1. 在system\_core2.js的selectable事件中添加初始化对应模块插件的方法：

//模块锁：只初始化一次模块插件

var userPanel=true;

var orderPanel=true;

$( ".selectable" ).selectable({

stop: function() {

$( ".ui-selected", this ).each(function(index, domEle) {

//alert( $(domEle).attr('id'));

$(".panel").hide();

var panelId = $(domEle).attr('id')+"Panel";

$("#"+panelId).show();

//添加点击后初始化对应模块的插件：

switch(panelId){

case "userPanel":

if(userPanel){//永远只渲染一次

new userPanelGrid2().init();

userPanel = false;

}

break;

case "orderPanel":

if(orderPanel){ //永远只渲染一次

//init

orderPanel = false;

}

break;

}

});

}

});

1. 测试：如果成功初始化，并且只初始化一次则提示一次init。

new userPanelGrid2().init();对应上述userPanelGrid2.js文件的初始化方法：

userPanelGrid2= function(){

return {

init : function(){

alert("init");

//grid

}

};

};

1. 把grid插件代码放到上述//grid的位置。

### userPanelGrid2.js

userPanelGrid2= function(){

return {

init : function(){

//grid 只会在new之后调用init才进行插件的渲染

//alert("init");

$("#userPanelGrid").jqGrid({

datatype: "local",

//JSON数据格式

// url:'findAll.action', datatype: "json",

height: 250,

//autoheight: true,

width:window.screen.availWidth-265,

autowidth: false,

shrinkToFit: true,

colNames:['Inv No','User Name', 'Password'],

colModel:[

{name:'id',index:'id', width:60, sorttype:"int"},

{name:'userName',index:'userName', width:100},

{name:'password',index:'password', width:100}

],

pager: '#userPanelGridPager',

rowNum:50,

rowList:[10,20,30,40,50],

viewrecords: true,

multiselect: true,

sortname: 'id',

sortorder: "desc",

hidegrid: false,

caption: "User Panel Grid"

});

}

};

};

### 用户体验：样式优化

### jquery-ui.css

.ui-accordion .ui-accordion-content {

padding: 1em 2.2em;

border-top: 0;

overflow: auto;

}

.ui-tabs .ui-tabs-panel {

display: block;

border-width: 0;

padding: 1em 1.4em;

background: none;

}

### system\_core2.js

$(window).resize(function(){

$(window).unbind("onresize");

//window resize后重新设置所有grid的宽度

$(".panelGrid").setGridWidth($(window).width() - 236);

$(".panelGrid").setGridHeight($(window).height() - 200);

$(window).bind("onresize", this);

});

### userPanelGrid2.js

$("#userPanelGrid").jqGrid({

width:window.screen.availWidth-236,

height: window.screen.availHeight-320,

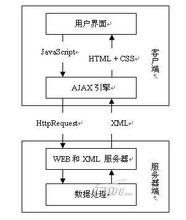
//,caption: "User Panel Grid"

自定义JQuery UI Theme :

<http://jqueryui.com/themeroller/>

# AJAX技术

AJAX 工作原理

[](http://baike.baidu.com/picture/1641/5762264/0/b21c8701a18b87d61337414f050828381f30fd6d.html?fr=lemma&ct=single)

Ajax（Asynchronous JavaScript + XML）的定义 ，包含的技术；

1. 基于web标准（standards-based presentation）XHTML+CSS的表示；
2. 使用 DOM（Document Object Model）进行动态显示及交互；
3. 使用 XML 和 XSLT 进行数据交换及相关操作；
4. 使用 XMLHttpRequest 进行异步数据查询、检索；
5. 使用 JavaScript 将所有的东西绑定在一起。
6. 类似于DHTML或LAMP，AJAX不是指一种单一的技术，而是有机地利用了一系列相关的技术。事实上，一些基于AJAX的“派生/合成”式（derivative/composite）的技术正在出现，如“AFLAX”。   
     
   AJAX的应用使用支持以上技术的web浏览器作为运行平台。这些浏览器目前包括：Mozilla、Firefox、Internet Explorer、Opera、Konqueror及Safari。但是Opera不支持XSL格式对象，也不支持XSLT。
7. AJAX即“Asynchronous Javascript And XML”（异步JavaScript和XML），是指一种创建交互式[网页](http://baike.baidu.com/view/828.htm" \t "_blank)应用的网页开发技术。
8. AJAX = 异步 [JavaScript](http://baike.baidu.com/view/16168.htm)和[XML](http://baike.baidu.com/view/63.htm" \t "_blank)（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)的子集）。
9. AJAX 是一种用于创建快速动态网页的技术。AJAX 可使因特网应用程序更小、更快，更友好。
10. 通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。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。
11. 传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页面。
12. Ajax 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。简而言之，XmlHttpRequest 使您可以使用 JavaScript 向服务器提出请求并处理响应，而不阻塞用户。
13. AJAX 是一种独立于 Web 服务器软件的浏览器技术。AJAX 基于下列 Web 标准：
    1. JavaScript [XML](http://baike.baidu.com/view/63.htm) [HTML](http://baike.baidu.com/view/692.htm) [CSS](http://baike.baidu.com/view/15916.htm)在 AJAX 中使用的 Web 标准已被良好定义，并被所有的主流浏览器支持。AJAX 应用程序独立于浏览器和平台。
    2. Web 应用程序较桌面应用程序有诸多优势；它们能够涉及广大的用户，它们更易安装及维护，也更易开发。

## AJAX - 浏览器支持

AJAX ：不同的浏览器创建 XMLHttpRequest 对象的方法是有差异的。

XMLHttpRequest的使用：

1. 创建XMLHttpRequest对象
2. IE 浏览器使用 ActiveXObject。不同IE版本创建ActiveXObject对象的参数不一。

var xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

var xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

1. 而其他的浏览器使用名为 XMLHttpRequest 的 JavaScript 内建对象。

// Firefox, Opera 8.0+, Safari

var xmlHttp=new XMLHttpRequest();

1. 使用XMLHttpRequest对象与服务器进行通信。
   1. 使用XMLHttpRequest向服务器发送一个请求
   2. url？userName=wenyuxinsx@qq.com

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

xmlHttp.send(null);

* 1. onreadystatechange 属性存有处理服务器响应的函数。（监听服务器响应状态）

xmlHttp.onreadystatechange=function()

{

// 我们需要在这里写一些代码

}

* 1. readyState 属性

|  |  |
| --- | --- |
| **状态** | **描述** |
| 0 | 请求未初始化（在调用 open() 之前） |
| 1 | 请求已提出（调用 send() 之前） |
| 2 | 请求已发送（这里通常可以从响应得到内容头部） |
| 3 | 请求处理中（响应中通常有部分数据可用，但是服务器还没有完成响应） |
| 4 | 请求已完成（可以访问服务器响应并使用它） |

xmlHttp.onreadystatechange=function()

{

if(xmlHttp.readyState==4){//处理数据}

}

* 1. responseText 属性和responseXml属性

xmlHttp.onreadystatechange=function()

{

if(xmlHttp.readyState==4){

//处理数据

xmlHttp.responseText

//xmlHttp.responseXml

//JS解析JSON数据格式

}

}

**jQuery的AJAX**

$.ajax( {

url:'customerManagement/deleteCustomer.action',

data : {

selectNodeId : ids

},

type : 'POST',

dataType : 'json',

success : function() {

$("#personalCustomerGrid").trigger("reloadGrid");

}

});

## JQuery中$.ajax()方法参数详解

url: 要求为String类型的参数，（默认为当前页地址）发送请求的地址。

type: 要求为String类型的参数，请求方式（post或get）默认为get。注意其他http请求方法，例如put和

      delete也可以使用，但仅部分浏览器支持。

timeout: 要求为Number类型的参数，设置请求超时时间（毫秒）。此设置将覆盖$.ajaxSetup()方法的全局设

         置。

async：要求为Boolean类型的参数，默认设置为true，所有请求均为异步请求。

       如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其他操作必须等

       待请求完成才可以执行。

cache：要求为Boolean类型的参数，默认为true（当dataType为script时，默认为false）。

       设置为false将不会从浏览器缓存中加载请求信息。

data: 要求为Object或String类型的参数，发送到服务器的数据。如果已经不是字符串，将自动转换为字符串格

      式。get请求中将附加在url后。防止这种自动转换，可以查看processData选项。对象必须为key/value格

      式，例如{foo1:"bar1",foo2:"bar2"}转换为&foo1=bar1&foo2=bar2。如果是数组，JQuery将自动为不同

      值对应同一个名称。例如{foo:["bar1","bar2"]}转换为&foo=bar1&foo=bar2。

dataType: 要求为String类型的参数，预期服务器返回的数据类型。如果不指定，JQuery将自动根据http包mime

          信息返回responseXML或responseText，并作为回调函数参数传递。

          可用的类型如下：

xml：返回XML文档，可用JQuery处理。

          html：返回纯文本HTML信息；包含的script标签会在插入DOM时执行。

          script：返回纯文本JavaScript代码。不会自动缓存结果。除非设置了cache参数。注意在远程请求

                  时（不在同一个域下），所有post请求都将转为get请求。

          json：返回JSON数据。

          jsonp：JSONP格式。使用SONP形式调用函数时，例如myurl?callback=?，JQuery将自动替换后一个

                “?”为正确的函数名，以执行回调函数。

          text：返回纯文本字符串。

beforeSend：要求为Function类型的参数，发送请求前可以修改XMLHttpRequest对象的函数，例如添加自定义

            HTTP头。在beforeSend中如果返回false可以取消本次ajax请求。XMLHttpRequest对象是惟一的参

            数。

            function(XMLHttpRequest){

               this;   //调用本次ajax请求时传递的options参数

            }

complete：要求为Function类型的参数，请求完成后调用的回调函数（请求成功或失败时均调用）。

          参数：XMLHttpRequest对象和一个描述成功请求类型的字符串。

          function(XMLHttpRequest, textStatus){

             this;    //调用本次ajax请求时传递的options参数

          }

success：要求为Function类型的参数，请求成功后调用的回调函数，有两个参数。

         (1)由服务器返回，并根据dataType参数进行处理后的数据。

         (2)描述状态的字符串。

         function(data, textStatus){

            //data可能是xmlDoc、jsonObj、html、text等等

            this;  //调用本次ajax请求时传递的options参数

error：要求为Function类型的参数，请求失败时被调用的函数。该函数有3个参数，即XMLHttpRequest对象、错

       误信息、捕获的错误对象(可选)。

       ajax事件函数如下：

       function(XMLHttpRequest, textStatus, errorThrown){

          //通常情况下textStatus和errorThrown只有其中一个包含信息

          this;   //调用本次ajax请求时传递的options参数

       }

contentType：要求为String类型的参数，当发送信息至服务器时，内容编码类型默认

             为"application/x-www-form-urlencoded"。该默认值适合大多数应用场合。

dataFilter：要求为Function类型的参数，给Ajax返回的原始数据进行预处理的函数。

            提供data和type两个参数。data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的

            dataType参数。函数返回的值将由jQuery进一步处理。

            function(data, type){

                //返回处理后的数据

                return data;

            }

global：要求为Boolean类型的参数，默认为true。表示是否触发全局ajax事件。设置为false将不会触发全局

        ajax事件，ajaxStart或ajaxStop可用于控制各种ajax事件。

ifModified：要求为Boolean类型的参数，默认为false。仅在服务器数据改变时获取新数据。

            服务器数据改变判断的依据是Last-Modified头信息。默认值是false，即忽略头信息。

jsonp：要求为String类型的参数，在一个jsonp请求中重写回调函数的名字。

       该值用来替代在"callback=?"这种GET或POST请求中URL参数里的"callback"部分，例如

       {jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。

username：要求为String类型的参数，用于响应HTTP访问认证请求的用户名。

password：要求为String类型的参数，用于响应HTTP访问认证请求的密码。

processData：要求为Boolean类型的参数，默认为true。默认情况下，发送的数据将被转换为对象（从技术角度

             来讲并非字符串）以配合默认内容类型"application/x-www-form-urlencoded"。如果要发送DOM

             树信息或者其他不希望转换的信息，请设置为false。

scriptCharset：要求为String类型的参数，只有当请求时dataType为"jsonp"或者"script"，并且type是GET时

               才会用于强制修改字符集(charset)。通常在本地和远程的内容编码不同时使用。

案例代码：

$(function(){

    $('#send').click(function(){

         $.ajax({

             type: "GET",

             url: "test.json",

             data: {username:$("#username").val(), content:$("#content").val()},

             dataType: "json",

             success: function(data){

                         $('#resText').empty();   //清空resText里面的所有内容

                         var html = '';

                         $.each(data, function(commentIndex, comment){

                               html += '<div class="comment"><h6>' + comment['username']

                                         + ':</h6><p class="para"' + comment['content']

                                         + '</p></div>';

                         });

                         $('#resText').html(html);

                      }

         });

    });

});

顺便说一下$.each()函数:

$.each()函数不同于JQuery对象的each()方法，它是一个全局函数，不操作JQuery对象，而是以一个数组或者对象作为第1个参数，以一个回调函数作为第2个参数。回调函数拥有两个参数：第1个为对象的成员或数组的索引，第2个为对应变量或内容。

**关于jQuery的AJAX不兼容IE的解决办法**

在使用jQuery的AJAX：get方法去检测数据是否存在时，会发现IE会出现不兼容的情况。

用AJAX:post方法时，使用Chrome/FireFox/IE均能出现正确的结果，但是在使用AJAX:get方法时，IE却不能返回正确的结果。

难道是数据超出了get方法的限制的长度，这个也不可能，我总共才传了一点点数据。排除。

网上一些网友说是IE缓存的问题，在请求数据后边加上随机数就行，比如加上时间数new Date().getTime()。

先前的代码中我已经添加了随机数，用的是“Math.random()”也不行。难道用时间比较靠谱？

那就改成获取时间试试，在参数后加“new Date().getTime()”后反复测试还是不行，真是百思不得其解！这个错误也排除了。

反复查看手册后发现请求的数据格式还是有一种JSON格式，如{foo:["bar1", "bar2"]} ，然后就按照这种格式书写，还真的返回了正确的查询结果。真不知道IE还有这点要求。（完）

先前的格式：

**type: "get",**

**data: "bid="+my\_bid+"&name\_cn="+name\_cn+"&timeStamp="+new Date().getTime(),**

改进后格式：

**type: "get",**

**data: {'bid':my\_bid,'name\_cn':name\_cn,'timeStamp':new Date().getTime()},**

在jQuery手册中是这样描述的：

*data Object,String*

*发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。*

*查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。*

*如果为数组，jQuery 将自动为不同值对应同一个名称。如 {foo:["bar1", "bar2"]} 转换为 "&foo=bar1&foo=bar2"。*

**代码片段:**

var siteUrl="http://blog.sina.com.cn/cnwyt";

jQuery.ajax({

**type: "get",**

url: siteUrl+"cosmetics/product/ajax\_check?",

//data: "bid="+my\_bid+"&name\_cn="+name\_cn+"&timeStamp=" + new Date().getTime(),

**data: {'bid':my\_bid,'name\_cn':name\_cn,'timeStamp':new Date().getTime()},**

dataType: 'json',

error: function (err) { alert('网络故障,请与管理员联系！') },

success: function (message) {

if(message!=false){

//ture的代码

}else{

//false的代码

}

});

# JSON与XML