# jQuery UI 概述

## jQuery UI

jQuery UI 是以 jQuery 为基础的开源 JavaScript 网页用户界面代码库。包含底层用户交互、动画、特效和可更换主题的可视控件。我们可以直接用它来构建具有很好交互性的 web 应用程序。

jQuery UI 的官网网站为：http://jqueryui.com/，我们下载最新版本的即可。目前本课采用的最新版本为：jquery-ui-1.10.3.custom.zip。里面目录结构如下：

1.css，包含与 jQuery UI 相关的 CSS 文件；

2.js，包含 jQuery UI 相关的 JavaScript 文件；

3.Development-bundle，包含多个不同的子目录：demos(jQuery UI 示例文件)、docs(jQueryUI 的文档文件)、themes(CSS 主题文件)和 ui(jQuery ui 的 JavaScript 文件)。

4.Index.html，可以查看 jQuery UI 功能的索引页。

## CSS 主题

CSS 主题就是 jQuery UI 的皮肤，有各种色调的模版提供使用。对于程序员，可以使用最和网站符合的模版；对于美工，也提供了没有任何样式的模版基于设计。我们可以在这里：http://jqueryui.com/themeroller/查看已有模版样式

# 创建 header 区

## 创建界面

我们首先要设计一个 header，这个区域将要设计成永远置顶。也就是，往下拉出滚动条也永远在页面最上层可视区内。在 header，目前先设计 LOGO、搜索框、按钮、注册和登录即可。

## 引入 jQuery UI

在目前的这个 header 区域中，有两个地方使用了 jQuery UI。一个是 button 按钮，一个是 dialog 对话框。

```
//JS 引入和 CSS 引入

<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/jquery.ui.js"></script>
<script type="text/javascript" src="js/index.js"></script>
<link rel="shortcut icon" type="image/x-icon" href="img/favicon.ico" />
<link rel="stylesheet" href="css/smoothness/jquery.ui.css" type="text/css" />
<link rel="stylesheet" href="css/style.css" type="text/css" />

//HTML 代码
<div id="header">
        <div class="header_main">
              <h1>知问</h1>
              <div class="header_search">
                    <input type="text" name="search" class="search" />
              </div>
              <div class="header_button">
                    <input type="button" value="查询" id="search_button" />
              </div>
              <div class="header_member">
                    <a href="###" id="reg_a">注册</a> |<a
          href="javascript:void(0)" id="login_a">登录</a>
              </div>
        </div>
</div>
<div id="reg" title="会员注册">
          表单区
</div>
```

```
//CSS
body {
    margin:0;
    padding:0;
    font-size:12px;
    margin:40px 0 0 0;
    background:#fff;
}
#header {
    height:40px;
    width:100%;
    background:url(../img/header_bg.png);
    position:absolute;
    top:0;
}
#header .header_main {
    width:800px;
    height:40px;
    margin:0 auto;
}
#header .header_main h1 {
    height:40px;
    line-height:40px;
    font-size:20px;
    color:#666;
    margin:0;
    padding:0 10px;
    float:left;
}
#header .header_search {
    float:left;
    padding:6px 0 0 0;
}
#header .header_search .search {
    width:300px;
    height:24px;
    border:1px solid #ccc;
    background:#fff;
    font-size:14px;
    color:#666;
    text-indent:5px;
}
#header .header_button {
    float:left;
    padding:5px;
}
#header .header_member {
    float:right;
    height:40px;
    line-height:40px;
```

```
}
#header .header_member a {
    font-size:14px;
    text-decoration:none;
    color:#555555;
}
```

# 对话框 UI

对话框（dialog），是 jQuery UI 非常重要的一个功能。它彻底的代替了 JavaScript 的 alert()、prompt()等方法，也避免了新窗口或页面的繁杂冗余。

我们可以同时打开多个 dialog，只要设置不同的 id 即可实现：

```
//调用 dialog 方法即可
$('#reg').dialog();
$('#login').dialog();
```

## 修改 dialog 样式

在弹出的 dialog 对话框中，在火狐浏览器中打开 Firebug 或者右击->查看元素。这样，我们可以看看 dialog 的样式，根据样式进行修改。我们为了和网站主题符合，对 dialog 的标题背景进行修改。

```
//无须修改 ui 里的 CSS，直接用 style.css 替代掉
.ui-widget-header {
        background:url(../img/ui_header_bg.png);
}
```

## dialog()方法的属性

对话框方法有两种形式：

1. dialog(options)，options 是以**对象键值对**的形式传参，每个键值对表示一个选项；
2. dialog('action', param)，action 是操作对话框方法的字符串，param 则是 options 的某个选项。

## dialog 参数说明

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| title | 无/字符串 | 对话框的标题，可以直接设置在 DOM 元素上 |
| buttons | 无/对象 | 以对象键值对方式，给 dialog 添加按钮，键是按钮的名称，值是用户点击后调用的回调函数 |

```
//例如：
$('#reg').dialog({
        title : '注册知问',
        buttons : {
        '按钮' : function () {}
        }
});
```

## dialog 位置属性

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| Position | center | 设置一个对话框窗口的坐标位置 center。 |

其他设置值为：left top、top right、bottom left、
position center/字符串 right bottom（四个角）、top、bottom（顶部或底部，宽度居中）、left 或 right（左边或右边，高度居中）、center（默认值）

```
$('#reg').dialog({
      position : 'left top'
});
```

## dialog 大小属性

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| width | 300/数值 | 对话框的宽度。默认为 300，单位是像素。 |
| Height | auto/数值 | 对话框的高度。默认为 auto，单位是像素。 |
| minWidth | 150/数值 | 对话框的最小宽度。默认 150，单位是像素。 |
| minHeight | 150/数值 | 对话框的最小高度。默认 150，单位是像素。 |
| maxWidth | auto/数值 | 对话框的最大宽度。默认 auto，单位是像素。 |
| maxHeight | auto/数值 | 对话框的最大高度。默认 auto，单位是像素。 |

```
//代码示例：
$('#reg').dialog({
    height : 500,
    width : 500,
    minWidth : 300,
    minHeight : 300,
    maxWidth : 800,
    maxHeight : 600
});
```

## dialog 显示隐藏

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| Show | false/布尔值 | 显示对话框时，默认采用淡入效果。 |
| Hide | false 布尔值 | 关闭对话框时，默认采用淡出效果。 |

### show 和 hide 可选特效

| 特效名称 | 说明 |
| --- | --- |
| blind | 对话框从顶部显示或消失 |
| bounce | 对话框断断续续地显示或消失，垂直运动 |
| clip | 对话框从中心垂直地显示或消失 |
| slide | 对话框从左边显示或消失 |
| drop | 对话框从左边显示或消失，有透明度变化 |
| fold | 对话框从左上角显示或消失 |
| highlight | 对话框显示或消失，伴随着透明度和背景色的变化 |
| puff | 对话框从中心开始缩放。显示时“收缩”，消失时“生长” |
| scale | 对话框从中心开始缩放。显示时“生长”，消失时“收缩” |
| pulsate | 对话框以闪烁形式显示或消失 |

//示例代码

```
$('#reg').dialog({
    show : true,
    hide : true
});
```

注意：设置 true 后，默认为淡入淡出，

如果想使用别的特效，可以使用以下表格中的字符串参数。

```
$('#reg').dialog({
    show : 'blind',
    hide : 'blind'
});
```

## dialog 行为属性

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| autoOpen | true/布尔值 | 默认为 true，调用 dialog()方法时就会打开对话框；如果为 false，对话框不可见，但对话框已创建，可以通过 dialog('open')才能可见。 |
| draggable | true/布尔值 | 默认为 true，可以移动对话框，false 无法移动。 |
| resizable | True/布尔值 | 默认为 true，可以调整对话框大小，false 无法调整 |
| modal | false/布尔值 | 默认为 false，对话框外可操作，true 对话框会遮罩一层灰纱，无法操作。 |
| closeText | 无/字符串 | 设置关闭按钮的 title 文字 |

```
//示例代码
$('#reg').dialog({
    autoOpen : false,
    draggable : false,
    resizable : false,
    modal : true,
    closeText : '关闭'
});
```

## dialog()方法的事件

除了属性设置外，dialog()方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的 this 值等于对话框内容的 div 对象，不是整个对话框的 div。

## dialog 事件

| 事件名 | 说明 |
| --- | --- |
| focus | 当对话框被激活时（首次显示以及每次在上面点击）会调用 focus 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| create | 当对话框被创建时会调用 create 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| open | 当对话框被显示时（首次显示或调用 dialog('open')方法）会调用 open 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| beforeClose | 当对话框将要关闭时（当单击关闭按钮或调用 dialog('close')方法），会调用 beforeclose 方法。如果该函数返回 false，对话框将不会被关闭。关闭的对话框可以用 dialog('open')重新打开。该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| close | 当对话框将要关闭时（当单击关闭按钮或调用 dialog('close')方法），会调用 close 方法。关闭的对话框可以用 dialog('open')重新打开。该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| drag | 当对话框移动时，每次移动一点均会调用 drag 方法。该方法有两个参数。该方法有两个参数(event, ui)。此事件中的 ui 有两个属性对象：<br>1.position，得到当前移动的坐标，有两个子属性：top 和 left。<br>2.offset，得到当前移动的坐标，有两个子属性：top 和 left。 |
| dragStart | 当开始移动对话框时，会调用 dragStart 方法。该方法有两个参数(event, ui)。此事件中的 ui 有两个属性对象：<br>1.position，得到当前移动的坐标，有两个子属性：top 和 left。<br>2.offset，得到当前移动的坐标，有两个子属性：top 和 left。 |
| dragStop | 当开始移动对话框时，会调用 dragStop 方法。该方法有两个参数(event, ui)。此事件中的 ui 有两个属性对象：<br>1.position，得到当前移动的坐标，有两个子属性：top 和 left。<br>2.offset，得到当前移动的坐标，有两个子属性：top 和 left。 |

| | |
|---|---|
| resize | 当对话框拉升大小的时候，每一次拖拉都会调用 resize 方法。该方法有两个参数(event, ui)。此事件中的 ui 有四个属性对象：<br>1.size，得到对话框的大小，有两个子属性：width 和 height。<br>2.position，得到对话框的坐标，有两个子属性：top 和 left。<br>3.originalSize，得到对话框原始的大小，有两个子属性：width 和 height。<br>4.originalPosition，得到对话框原始的坐标，有两个子属性：top 和 left。 |
| resizeStart | 当开始拖拉对话框时，会调用 resizeStart 方法。该方法有两个参数(event, ui)。此事件中的 ui 有四个属性对象：<br>1.size，得到对话框的大小，有两个子属性：width 和 height。<br>2.position，得到对话框的坐标，有两个子属性：top 和 left。<br>3.originalSize，得到对话框原始的大小，有两个子属性：width 和 height。<br>4.originalPosition，得到对话框原始的坐标，有两个子属性：top 和 left。 |
| resizeStop | 当结束拖拉对话框时，会调用 resizeStart 方法。该方法有两个参数(event, ui)。此事件中的 ui 有四个属性对象：<br>1.size，得到对话框的大小，有两个子属性：width 和 height。<br>2.position，得到对话框的坐标，有两个子属性：top 和 left。<br>3.originalSize，得到对话框原始的大小，有两个子属性：width 和 height。<br>4.originalPosition，得到对话框原始的坐标，有两个子属性：top 和 left。 |

## 示例代码：

```
//当对话框获得焦点后
$('#reg').dialog({
    focus : function (e, ui) {
    alert('获得焦点');
}
});
//当创建对话框时
$('#reg').dialog({
```

```
        create : function (e, ui) {
        alert('创建对话框');
        }
});//当将要关闭时
$('#reg').dialog({
        beforeClose : function (e, ui) {
        alert('关闭前做的事！');
        return flag;
}
});
//关闭对话框时
$('#reg').dialog({
        close : function (e, ui) {
        alert('关闭！');
}
});
//对话框移动时
$('#reg').dialog({
        drag : function (e, ui) {
                alert('top:' + ui.position.top + '\n'
                + 'left:' + ui.position.left);
        }
});
//对话框开始移动时
$('#reg').dialog({
dragStart : function (e, ui) {
alert('top:' + ui.position.top + '\n'
+ 'left:' + ui.position.left);
}
});
//对话框结束移动时
$('#reg').dialog({
        dragStop : function (e, ui) {
                alert('top:' + ui.position.top + '\n'
                + 'left:' + ui.position.left);
        }
});
//调整对话框大小时
$('#reg').dialog({
        resize : function (e, ui) {
```

```
        alert('size:' + ui.size.width + '\n'+ 'originalSize:' + ui.originalSize.width);
        }
});
//开始调整对话框大小时
$('#reg').dialog({
resizeStart : function (e, ui) {
alert('size:' + ui.size.width + '\n'
+ 'originalSize:' + ui.originalSize.width);
}
});
//结束调整对话框大小时
$('#reg').dialog({
resizeStop : function (e, ui) {
alert('size:' + ui.size.width + '\n'
+ 'originalSize:' + ui.originalSize.width);
}
});
```

## dialog('action', param)方法

| 方法 | 返回值 | 说明 |
| --- | --- | --- |
| dialog('open') | jQuery 对象 | 打开对话框 |
| dialog('close') | jQuery 对象 | 关闭对话框 |
| dialog('destroy') | jQuery 对象 | 删除对话框，直接阻断了 dialog。 |
| dialog('isOpen') | 布尔值 | 判断对话框是否打开状态 |
| dialog('widget') | jQuery 对象 | 获取对话框的 jQuery 对象 |
| dialog('option', param) | 一般值 | 获取 options 属性的值 |
| dialog('option', param, value) | jQuery 对象 | 设置 options 属性的值 |

```
//初始隐藏对话框
$('#reg').dialog({
autoOpen : false
});
//打开对话框
$('#reg_a').click(function () {
$('#reg').dialog('open');
});
```

```
//关闭对话框$('#reg').click(function () {
$('#reg').dialog('close');
});
//判断对话框打开或关闭状态
$(document).click(function () {
alert($('#reg').dialog('isOpen'));
});
//将指定对话框置前
$(document).click(function () {
    $('#reg').dialog('moveToTop');
});
//获取某个 options 的 param 选项的值
var title = $('#reg').dialog('option', 'title');
alert(title);

//设置某个 options 的 param 选项的值
$('#reg').dialog('option', 'title', '注册知问');
```

## dialog 中使用 on()

在 dialog 的事件中，提供了使用 on()方法处理的事件方法。

| 特效名称 | 说明 |
| --- | --- |
| dialogfocus | 得到焦点时触发 |
| dialogopen | 显示时触发 |
| dialogbeforeclose | 将要关闭时触发 |
| dialogclose | 关闭时触发 |
| dialogdrag | 每一次移动时触发 |
| dialogdragstart | 开始移动时触发 |
| dialogdragstop | 移动结束后触发 |
| dialogresize | 每次调整大小时触发 |
| dialogresizestart | 开始调整大小时触发 |
| dialogresizestop | 结束调整大小时触发 |

```
$('#reg').on('dialogclose', function () {
 alert('关闭');
```

```
});
```

# 按钮（button）组件

按钮（button），可以给生硬的原生按钮或者文本提供更多丰富多彩的外观。它不单单可以设置按钮或文本，还可以设置单选按钮和多选按钮。

## 使用 button 按钮

使用 button 按钮 UI 的时候，不一定必须是 input 按钮形式，普通的文本也可以设置成 button 按钮。

```
$('#search_button').button();
```

## 修改 button 样式

在弹出的 button 对话框中，在火狐浏览器中打开 Firebug 或者右击->查看元素。这样，我们可以看看 button 的样式，根据样式进行修改。我们为了和网站主题符合，对 dialog 的标题背景进行修改。

```
//无须修改 ui 里的 CSS，直接用 style.css 替代掉
.ui-state-default, .ui-widget-content .ui-state-default, .ui-widget-header .ui-state-default
{
     background:url(../img/ui_header_bg.png);
}
.ui-state-active, .ui-widget-content .ui-state-active, .ui-widget-header .ui-state-active
 {
   background:url(../img/ui_white.png);
}
```

## button()方法的属性

**按钮方法有两种形式：**

1.button(options)，options 是以对象键值对的形式传参，每个键值对表示一个选项；

2.button('action', param)，action 是操作对话框方法的字符串，param 则是 options 的某个选项。

## Button 按钮属性说明

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| disabled | false/布尔值 | 默认为 false，设置为 true 时，按钮是非激活的。 |
| label | 无/字符串 | 对应按钮上的文字。如果没有，HTML 内容将被作为按钮的文字。 |
| Icons | 无/字符串 | 对应按钮上的图标。在按钮文字前面和后面都可以放置一个图标，通过对象键值对的方式完成：<br>{<br>primary : 'ui-icon-search',<br>secondary : 'ui-icon-search'<br>} |
| text | true/布尔值 | 当时设置为 false 时，不会显示文字，但必须指定一个图标。 |

```
//演示代码：
$('#search_button').button({
    disabled : false,
    icons : {
        primary : 'ui-icon-search',
    },
    label : '查找',
    text : false,
});
```

注意：对于 button 的事件方法，只有一个：create，当创建 button 时调用。

## button('action', param)

button('action', param)方法能设置和获取按钮。action 表示指定操作的方式。

| 方法 | 返回值 | 说明 |
| --- | --- | --- |
| button('disable') | jQuery 对象 | 禁用按钮 |
| button('enable') | jQuery 对象 | 启用按钮 |
| button('destroy') | jQuery 对象 | 删除按钮，直接阻断了 button。 |
| button('refresh') | jQuery 对象 | 更新按钮布局。 |
| button('widget') | jQuery 对象 | 获取对话框的 jQuery 对象 |
| button('option', param) | 一般值 | 获取 options 属性的值 |
| button('option', param, value) | jQuery 对象 | 设置 options 属性的值 |

```
//禁用按钮
$('#search_button').button('disable');

//启用按钮
$('#search_button').button('enable');

//删除按钮
$('#search_button').button('destroy');

//更新按钮，刷新按钮
$('#search_button').button('refresh');

//得到 button 的 jQuery 对象
$('#search_button').button('widget');

//得到 button 的 options 值
alert($('#search_button').button('option', 'label'));

//设置 button 的 options 值
$('#search_button').button('option', 'label', '搜索');
```

注意：对于 UI 上自带的按钮，比如 dialog 上的，我们可以通过 Firebug 查找得到 jQuery 对象。

```
$('#reg').parent().find('button').eq(1).button('disable');
```

## 单选框、复选框

button 按钮不但可以设置普通的按钮，对于单选框、复选框同样有效。

```
//HTML 单选框
  <label for="male">男
        <input type="radio" name="sex" value="male" id="male"/>
  </label>
 <label for="female">女
        <input type="radio" name="sex" value="female" id="female"/>
 </label>
```

```
//jQuery 单选框
$('#reg input[type=radio]').button();

//jQuery 单选框改
```

```
$('#reg').buttonset(); //HTML 部分做成一行即可
```

```
//HTML 复选框
<label for="red">红
    <input type="checkbox" name="color" value="red" id="red"/>
</label>
<label for="green">绿
    <input type="checkbox" name="color" value="green" id="green"/>
</label>
<label for="yellow">黄
    <input type="checkbox" name="color" value="yellow" id="yellow">
</label>
```

```
//jQuery 复选框
$('#reg input[type=radio]').button();

//jQuery 复选框改
$('#reg').buttonset();
```

# 创建注册表单

通过前面已学的 jQuery UI 部件，我们来创建一个注册表单。

## HTML 部分

```
<div id="reg" title="会员注册">
    <p>
      <label for="user">帐号：</label>
       <input type="text" name="user" class="text" id="user" title="请输入帐号，
       不小于 2 位！" />
      <span class="star">*</span>
   </p>
   <p>
     <label for="pass">密码：</label>
     <input type="text" name="pass" class="text" id="pass" title="请输入密码，不
            小于 6 位！" />
     <span class="star">*</span>
   </p>
    <p>
```

```
        <label for="email">邮箱：</label>
        <input type="text" name="email" class="text" id="email"
            title="请输入电子邮件！" />
        <span class="star">*</span>
    </p>
     <p>
        <label>性别：</label>
        <input type="radio" name="sex" id="male" checked="checked"/>
        <label  for="male">男</label>
        <input type="radio" name="sex" id="female" />
        <label for="female">女</label>
     </p>
      <p>
        <label for="date">生日：</label>
        <input type="text" name="date" readonly="readonly"
        class="text" id="date" />
      </p>
</div>
```

## CSS 部分

```
#reg {
        padding:15px;
}
#reg p {
        margin:10px 0;
        padding:0;
}
#reg p label {
        font-size:14px;
        color:#666;
}
#reg p .star {
        color:red;
}
#reg .text {
        border-radius:4px;
        border:1px solid #ccc;
        background:#fff;
        height:25px;
        width:200px;
```

```
        text-indent:5px;
        color:#666;
    }
```

## jQuery 部分

```
        $('#reg').dialog({
            autoOpen : true,
            modal : true,
            resizable : false,
            width : 320,
            height : 340,
            buttons : {
                '提交' : function () {}
                 },
        });
        $('#reg').buttonset();
        $('#date').datepicker();
        $('#reg input[title]').tooltip();
```

# 工具提示框组件

工具提示（tooltip），是一个非常实用的 UI。它彻底扩展了 HTML 中的 title 属性，让提示更加丰富，更加可控制，全面提升了用户体验。

## 调用 tooltip()方法

在调用 tooltip()方法之前，首先需要针对元素设置相应 title 属性。

**<input type="text" name="user" class="text" id="user" title="请输入帐号，不小于**

**2 位！" />**

**$('#user').tooltip();**

## 修改 tooltip()样式

在弹出的 tooltip 提示框后，在火狐浏览器中打开 Firebug 或者右击->查看元素。这样，我们可以看看 tooltip 的样式，根据样式进行修改。

**//无须修改 ui 里的 CSS，直接用 style.css 替代掉**

```
.ui-tooltip {
    color:red;
}
```

## tooltip()方法的属性

对话框方法有两种形式：

**1.tooltip(options)**，options 是以对象键值对的形式传参，每个键值对表示一个选项；

**2.tooltip('action', param)**，action 是操作对话框方法的字符串，param 则是 options 的某个选项。

| tooltip 属性说明 | | |
|---|---|---|
| **属性** | **默认值/类型** | **说明** |
| disabled | false/布尔值 | 设置为 true，将禁止显示工具提示。 |
| content | 无/字符串 | 设置 title 内容。 |
| items | 无/字符串 | 设置选择器以限定范围。 |
| tooltipClass | 无/字符串 | 引入 class 形式的 CSS 样式。 |

```
//代码演示如下：
$('[title]').tooltip({
    disabled : false,
    content : '改变文字',
    items : 'input',
    tooltipClass : 'reg_tooltip'
});
```

| tooltip 页面位置选项 | | |
|---|---|---|
| **属性** | **默认值/类型** | **说明** |
| Position | 无/对象 | 使用对象的键值对赋值，有两个属性：my 和 at 表示坐标。My 是以目标点左下角为基准，at 以 my 为基准。 |

```
//示例代码
$('#user').tooltip({
     position : {
          my : 'left center',
          at : 'right+5 center'
     }
```

});

| tooltip 动画选项 | | |
|---|---|---|
| 属性 | 默认值/类型 | 说明 |
| show | false/布尔值 | 显示对话框时，默认采用淡入效果。 |
| hide | false 布尔值 | 关闭对话框时，默认采用淡出效果。 |

```
//示例代码
$('#user').tooltip({
        show : false,
        hide : false,
});
```

注意：设置 true 后，默认为淡入淡出，如果想使用别的特效，可以使用以下表格中的字符串参数。

| show 和 hide 可选特效 | |
|---|---|
| 特效名称 | 说明 |
| blind | 工具提示从顶部显示或消失 |
| bounce | 工具提示断断续续地显示或消失，垂直运动 |
| clip | 工具提示从中心垂直地显示或消失 |
| slide | 工具提示从左边显示或消失 |
| drop | 工具提示从左边显示或消失，有透明度变化 |
| fold | 工具提示从左上角显示或消失 highlight 工具提示显示或消失，伴随着透明度和背景色的变化 |
| puff | 工具提示从中心开始缩放。显示时“收缩”，消失时“生长” |
| scale | 工具提示从中心开始缩放。显示时“生长”，消失时“收缩” |
| pulsate | 工具提示以闪烁形式显示或消失 |

```
//示例代码
$('#user').tooltip({
      show : 'blind',
      hide : 'blind',
});
```

| tooltip 行为选项 | | |
|---|---|---|
| 属性 | 默认值/类型 | 说明 |
| track | false/布尔值 | 设置为 true，能跟随鼠标移动。 |

```
//示例代码
```

```
$('#user').tooltip({
    track : true,
});
```

## tooltip()方法的事件

除了属性设置外，tooltip()方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的 this 值等于对话框内容的 div 对象，不是整个对话框的 div。

| tootip 事件 | |
|---|---|
| 事件名 | 说明 |
| Create | 当工具提示被创建时会调用 create 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| open | 当工具提示被显示时，会调用 open 方法，该方法有两个参数(event, ui)。此事件中的 ui 有一个参数 tooltip，返回是工具提示的 jQuery 对象。 |
| Close | 当工具提示关闭时，会调用 close 方法。关闭的工具提示可以用 tooltip('open')重新打开。该方法有两个参数(event,ui)。此事件中的 ui 有一个参数 tooltip，返回是工具提示的 jQuery 对象。 |

```
//当创建工具提示时
$('#user').tooltip({
    create : function () {
        alert('创建触发！');
    }
});

//当工具提示关闭时
$('#user').tooltip({
    close : function () {
        alert('关闭触发');
    }
});

//当工具提示打开时
$('#user').tooltip({
    open : function () {
        alert('打开触发');
```

```
    }
});
```

## tooltip('action', param)方法

| 方法 | 返回值 | 说明 |
| --- | --- | --- |
| tooltip('open') | jQuery 对象 | 打开工具提示 |
| tooltip('close') | jQuery 对象 | 关闭工具提示 |
| tooltip('disable') | jQuery 对象 | 禁用工具提示 |
| tooltip('enable') | jQuery 对象 | 启用工具提示 |
| tooltip('destroy') | jQuery 对象 | 删除工具提示，直接阻断了 tooltip。 |
| tooltip('widget') | jQuery 对象 | 获取工具提示的 jQuery 对象 |
| tooltip('option', param) | 一般值 | 获取 options 属性的值 |
| tooltip('option', param, value) | jQuery 对象 | 设置 options 属性的值 |

```
//打开工具提示
$('#user').tooltip('open');

//关闭工具提示
$('#user').tooltip('close');

//禁用工具提示
$('#user').tooltip('disable');

//启用工具提示
$('#user').tooltip('enable');

//删除工具提示
$('#user').tooltip('destroy');

//获取工具提示 jQuery 对象
$('#user').tooltip('widget');

//获取某个 options 的 param 选项的值
var title = $('#user').tooltip('option', 'content');
alert(title);

//设置某个 options 的 param 选项的值
```

**$('#reg').dialog('option', 'content', '提示内容');**

## tooltip()中使用 on()

在 tooltip 的事件中，提供了使用 on()方法处理的事件方法。on()方法触发的对话框事件

| 事件名称 | 说明 |
| --- | --- |
| **tooltipopen** | 显示时触发 |
| **tooltipclose** | 每一次移动时触发 |

```
//示例代码
$('#reg').on('tooltipopen', function () {
    alert('打开时触发！');
});
```

# 自动补全组件

自动补全（autocomplete），是一个可以减少用户输入完整信息的 UI 工具。一般在输入邮箱、搜索关键字等，然后提取出相应完整字符串供用户选择。

## 调用 autocomplete()方法

```
//示例代码
$('#email').autocomplete({
    source : ['aaa@163.com', 'bbb@163.com', 'ccc@163.com'],
});
```

## 修改 autocomplete()样式

由于 autocomplete()方法是弹窗，然后鼠标悬停的样式。我们通过 Firebug 想获取到悬停时背景的样式，可以直接通过 jquery.ui.css 里面找相应的 CSS。

```
//无须修改 ui 里的 CSS，直接用 style.css 替代掉
.ui-menu-item a.ui-state-focus {
    background:url(../img/ui_header_bg.png);
}
```

**注意：其他修改方案类似。**

## autocomplete()方法的属性

自动补全方法有两种形式：

**1. autocomplete(options)，options** 是以对象键值对的形式传参，每个键值对表示一个选项；

**2. autocomplete('action', param)，**action 是操作对话框方法的字符串，param 则是 options 的某个选项。

**autocomplete 属性说明**

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| disabled | false/布尔值 | 设置为 true，将禁止显示自动补全。 |
| source | 无/数组 | 指定数据源， 可以是本地的，也可以是远程的。 |
| minLength | 1/数值 | 默认为 1，触发补全列表最少输入字符数。 |
| delay | 300/数值 | 默认为 300 毫秒，延迟显示设置。 |
| autofocus | false/布尔值 | 设置为 true 时，第一个项目会自动被选定。 |

```
//示例代码
$('#email').autocomplete({
    source : ['aaa@163.com', 'bbb@163.com', 'ccc@163.com'],
    disabled : false,
    minLength : 2,
    delay : 50,
    autoFocus : true,
});
```

**autocomplete 页面位置选项**

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| Position | 无/对象 | 表示坐标。使用对象的键值对赋值，有两个属性：my 和 at，my 是以目标点左上角为基准，at 以目标点右下角为基准。 |

```
//示例代码
$('#email').autocomplete({
    position : {
        my : 'left center',
        at : 'right center'
    }
});
```

## autocomplete()方法的事件

除了属性设置外，autocomplete()方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的 this 值等于对话框内容的 div 对象，不是整个对话框的 div。

**autocomplete 事件**

| 事件名 | 说明 |
| --- | --- |
| create | 当自动补全被创建时会调用 create 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| open | 当自动补全被显示时，会调用 open 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| close | 当自动补全被关闭时，会调用 close 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| focus | 当自动补全获取焦点时，会调用 focus 方法，该方法有两个参数(event, ui)。此事件中的 ui 有一个子属性对象 item，分别有两个属性：label，补全列表显示的文本；value，将要输入框的值。一般 label 和 value 值相同。 |
| select | 当自动补全获被选定时，会调用 select 方法，该方法有两个参数(event, ui)。此事件中的 ui 有一个子属性对象 item，分别有两个属性：label，补全列表显示的文本；value，将要输入框的值。一般 label 和 value 值相同。 |
| change | 当自动补全失去焦点且内容发生改变时，会调用 change 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| search | 当自动补全搜索完成后，会调用 search 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数为空。 |
| response | 当自动补全搜索完成后，在菜单显示之前，会调用 response 方法，该方法有两个参数(event, ui)。此事件中的 ui 参数有一个子对象 content，他会返回 label 和 value 值，可通过遍历了解。 |

**//示例代码**

```
$('#email').autocomplete({
    source : ['aaa@163.com', 'bbb@163.com', 'ccc@163.com'],
    disabled : false,
    minLength : 1,
```

```
    delay : 0,
    focus : function (e, ui) {
        ui.item.value = '123';
    },
    select : function (e, ui) {
        ui.item.value = '123';
    },
    change : function (e, ui) {
        alert('');
    },
    search : function (e, ui) {
        alert('');
    },
});
```

## autocomplete('action', param)方法

| 方法 | 返回值 | 说明 |
| --- | --- | --- |
| autocomplete('close') | jQuery 对象 | 关闭自动补齐 |
| autocomplete('disable') | jQuery 对象 | 禁用自动补齐 |
| autocomplete('enable') | jQuery 对象 | 启用自动补齐 |
| autocomplete('destroy') | jQuery 对象 | 删除自动补齐，直接阻断。 |
| autocomplete('widget') | jQuery 对象 | 获取工具提示的 jQuery 对象 |
| autocomplete('search',value) | jQuery 对象 | 在数据源获取匹配的字符串。 |

```
//关闭自动补全
$('#email').autocomplete('close');

//禁用自动补全
$('#email').autocomplete('disable');

//启用自动补全
$('#email').autocomplete('enable');

//删除自动补全
$('#email').autocomplete('destroy');

//获取自动补全 jQuery 对象
$('#email').autocomplete('widget');
```

```
//设置自动补全 search
$('#email').autocomplete('search', '');

//获取某个 options 的 param 选项的值
var delay = $('#email').autocomplete('option', 'delay');
alert(delay);

//设置某个 options 的 param 选项的值
$('#email').dialog('option', 'delay', 0);
```

## autocomplete 中使用 on()

在 autocomplete 的事件中，提供了使用 on()方法处理的事件方法。

| on()方法触发的对话框事件 | |
|---|---|
| 事件名称 | 说明 |
| autocompleteopen | 显示时触发 |
| autocompleteclose | 关闭时触发 |
| autocompletesearch | 查找时触发 |
| autocompletefocus | 获得焦点时触发 |
| autocompleteselect | 选定时触发 |
| autocompletechange | 改变时触发 |
| autocompleteresponse | 搜索完毕后，显示之前 |

```
//示例代码
$('#reg').on('autocompleteopen', function () {
    alert('打开时触发！');
});
```

# 邮箱自动补全

本节课，我们通过自动补全 source 属性的 function 回调函数，来动态的设置我们的数据源，以达到可以实现邮箱补全功能。

## 数据源 function

自动补全 UI 的 source 不但可以是数组，也可以是 function 回调函数。提供了自带的两个参数设置动态的数据源。

```
//示例代码
$('#email').autocomplete({
    source : function (request, response) {
        alert(request.term); //可以获取你输入的值
        response(['aa', 'aaaa', 'aaaaaa', 'bb']); //展示补全结果
    },
});
```

**注意：**这里的 response 不会根据你搜索关键字而过滤无关结果，而是把整个结果全部呈现出来。因为 source 数据源，本身就是给你动态改变的，就由你自定义，从而放弃系统内置的搜索能力。

## 邮箱自动补全代码实现

```
$('#email').autocomplete({
    autoFocus : true,
    delay : 0,
    source : function (request, response) {
        //起始
        var hosts = ['qq.com','163.com', '263.com', 'gmail.com', 'hotmail.com'],
        term = request.term, //获取输入值
        ix = term.indexOf('@'), //@
        name = term, //用户名
        host = '', //域名
        result = []; //结果
        //结果第一条是自己输入
        result.push(term);if (ix > -1) { //如果有@的时候
        name = term.slice(0, ix); //得到用户名
        host = term.slice(ix + 1); //得到域名
    }
    if (name) {
```

```
            //得到找到的域名
            var findedHosts = (host ? $.grep(hosts, function (value, index) {
                    return value.indexOf(host) > -1;
                }) : hosts),
                //最终列表的邮箱
                findedResults = $.map(findedHosts, function (value, index) {
                    return name + '@' + value;
                });
            //增加一个自我输入
            result = result.concat(findedResults);
        }
        response(result);
    },
});
```

# 日历组件

日历（datepicker）UI，可以让用户更加直观的、更加方便的输入日期，并且还考虑不同国家的语言限制，包括汉语。

## 调用 datepicker()方法

```
$('#date').datepicker();
```

## 修改 datepicker()样式

日历 UI 的 header 背景和对话框 UI 的背景采用的是同一个 class，所以，在此之前已经被修改。所以，这里无须再修改了。

```
//无须修改 ui 里的 CSS，直接用 style.css 替代掉
.ui-widget-header {
    background:url(../img/ui_header_bg.png);
}

//修改当天日期的样式
.ui-datepicker-today .ui-state-highlight {
    border:1px solid #eee;
    color:#f60;
```

```
}

//修改选定日期的样式
.ui-datepicker-current-day .ui-state-active {
    border:1px solid #eee;
    color:#06f;
}
```

**注意：其他修改方案类似。**

# datepicker()方法的属性

日历方法有两种形式：

1.**datepicker(options)，options** 是以对象键值对的形式传参，每个键值对表示一个选项；

2.**datepicker('action', param)，action** 是操作对话框方法的字符串，param 则是 options 的某个选项。

## datepicker 国际化选项

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| dateFormat | mm/dd/yy/时间 | 指定日历返回的日期格式。 |
| dayNames | 英文日期/数组 | 以数组形式指定星期中的天的长格式。<br>比如：Sunday、Monday 等。<br>中文：星期日 |
| dayNamesShort | 英文日期/数组 | 以数组形式指定星期中的天的短格式。<br>比如：Sun、Mon 等。 |
| dayNamesMin | 英文日期/数组 | 以数组形式指定星期中的天的最小格式。<br>比如：Su、Mo 等。 |
| monthNames | 英文月份/数组 | 以数组形式指定月份的长格式名称（January、February 等）。<br>数组必须从 January 开始。 |
| monthNamesShort | 英文月份/数组 | 以数组形式指定月份的短格式名称（Jan、Feb 等）。数组必须从 January 开始。 |
| altField | 无/字符串 | 为日期选择器指定一个<input>域 |
| altFormat | 无/字符串 | 添加到<input>域的可选日期格式 |
| appendText | 无/字符串 | 在日期选择器的<input>域后面附加文本 |
| showWeek | false/布尔值 | 显示周 |
| weekHeader | 'Wk'/字符串 | 显示周的标题 |
| firstDay | 0/数值 | 指定日历中的星期从星期几开始。<br>0 表示星期日。 |

注意：默认情况下，日历显示为英文。如果你想使用中文日历，直接引入中文语言包即可。或者把中文语言包的几行代码整合到某个 js 文件里即可。

## 日期格式代码

| 代码 | 说明 |
| --- | --- |
| d | 月份中的天，从 1 到 31 |
| dd | 月份中的天，从 01 到 31 |
| o | 年份中的天，从 1 到 366 |
| oo | 年份中的天，从 001 到 366 |
| D | 星期中的天的缩写名称（Mon、Tue 等） |
| DD | 星期中的天的全写名称（Monday、Tuesday 等） |
| m | 月份，从 1 到 12 |
| mm | 月份，从 01 到 12 |
| M | 月份的缩写名称（Jan、February 等） |
| MM | 月份的全写名称（January、February 等） |
| y | 两位数字的年份（14 表示 2014） |
| yy | 四位数字的年份（2014） |
| @ | 从 01/01/1997 至今的毫秒数 |

```
//示例代码
$('#date').datepicker({
    dateFormat : 'yy-mm-dd',
    dayNames : ['星期日','星期一','星期二','星期三','星期四','星期五','星期六'],
    dayNamesShort : ['星期日','星期一','星期二','星期三','星期四','星期五','星期六'],
    dayNamesMin : ['日','一','二','三','四','五','六'],
    monthNames : ['一月','二月','三月','四月','五月',
                  '六月','七月','八月','九月','十月','十一月','十二月'],
    monthNamesShort : ['一','二','三','四','五','六','七','八','九','十','十一','十二'],
    altField : '#abc',
    altFormat : 'yy-mm-dd',
    appendText : '(yy-mm-dd)',
    firstDay : 1,
    showWeek : true,
    weekHeader : '周',
});
```

## datepicker 属性说明

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |

| | | |
|---|---|---|
| disabled | false/布尔值 | 禁用日历 |
| numberOfMonths | 1/数值日历中 | 同时显示的月份个数。默认为 1，如果设置 3 就同时显示 3 个月份。也可以设置数组：[3,2]，3 行 2 列共 6 个。 |
| showOtherMonths | false/布尔值 | 如果设置为 true，当月中没有使用的单元格会显示填充，但无法使用。默认为 false，会隐藏无法使用的单元格。 |
| selectOtherMonths | false/布尔值 | 如果设置为 true，表示可以选择上个月或下个月的日期。前提是 showOtherMonths 设置 true。 |
| changeMonth | false/布尔值 | 如果设置为 true，显示快速选择月份的下拉列表。 |
| changeYear | false/布尔值 | 如果设置为 true，显示快速选择年份的下来列表。 |
| isRTL | false/布尔值 | 是否由右向左绘制日历。 |
| autoSize | false/布尔值 | 是否自动调整控件大小，以适应当前的日期格式的输入 |
| showOn | 'focus'/字符串 | 默认值为 focus，获取焦点触发，还有 button 点击按钮触发和 both 任一事件发生时触发。 |
| buttonText | '...'/字符串 | 触发按钮上显示的文本 |
| buttonImage | 无/字符串 | 图片按钮地址 |
| buttonImageOnly | false/布尔值 | 设置为 true 则会使图片代替按钮 |
| showButtonPanel | false/布尔值 | 开启显示按钮面板 |
| closeText | 'done'/字符串 | 设置关闭按钮的文本 |
| currentText | 'Today'/字符串 | 设置获取今日日期的按钮文本 |
| nextText | 'Next'/字符串 | 设置下一月的 alt 文本 |
| prevText | 'Prev'/字符串 | 设置上一月的 alt 文本 |
| navigationAsDateFormat | false/字符串 | 设置 prev、next 和 current 的文字可以是 format 的日期格式。 |
| yearSuffix | 无/字符串 | 附加在年份后面的文本 |
| showMonthAfterYear | false/布尔值 | 设置为 true，则将月份放置在年份之后 |

```
//示例代码
$('#date').datepicker({
    disabled : true,
    numberOfMonths : [3,2],
```

```
    showOtherMonths : true,
    selectOtherMonths : true,
    changeMonth : true,
    changeYear : true,
    isRTL : true,
    autoSize : true,
    showButtonPanel: true,
    closeText : '关闭',
    currentText : '今天',
    showMonthAfterYear: true,
});

//示例代码
$('#date').datepicker({
    yearRange: '1950:2020',
    minDate : -10000,
    maxDate : 0, //可以用 new Date(2007,1,1)
    defaultDate : -1, //可以用'1m+3'
    hideIfNoPrevNext : true,
    gotoCurrent : false,
});
```

## 选择日期的字符串表示方法

| 属性 | 说明 |
| --- | --- |
| X | 当前日期之后的 x 天（其中 x 范围从 1 到 n）比如：1，2 |
| -x | 当前日期之前的 x 天（其中 x 范围从 1 到 n）比如：-1，-2 |
| Xm | 当前日期之后的 x 个月（其中 x 范围从 1 到 n）比如：1m，2m |
| -xm | 当前日期之前的 x 个月（其中 x 范围从 1 到 n）比如：-1m，-2m |
| xw | 当前日期之后的 x 周（其中 x 范围从 1 到 n）比如：1w，2w |
| -xw | 当前日期之后的 x 周（其中 x 范围从 1 到 n）比如：-1w，2w |

## datepicker 属性说明

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| showAnim | fadeIn/字符串 | 设置 false，无效果。默认效果 fadeIn。 |
| duration | 300/数值 | 日历显示或消失时的持续时间，单位毫秒。 |

```
//示例代码
$('#date').datepicker({
   yearRange: '1950:2020',
```

```
    showAnim : false,
    duration : 300,
});
```

## datepicker 可选特效

| 特效名称 | 说明 |
| --- | --- |
| blind | 日历从顶部显示或消失 |
| bounce | 日历断断续续地显示或消失，垂直运动 |
| clip | 日历从中心垂直地显示或消失 slide 日历从左边显示或消失 |
| drop | 日历从左边显示或消失，有透明度变化 |
| fold | 日历从左上角显示或消失 |
| highlight | 日历显示或消失，伴随着透明度和背景色的变化 |
| puff | 日历从中心开始缩放。显示时“收缩”，消失时“生长” |
| scale | 日历从中心开始缩放。显示时“生长”，消失时“收缩” |
| pulsate | 日历以闪烁形式显示或消失 |
| fadeIn | 日历显示或消失时伴随透明度变化 |

# datepicker()方法的事件

除了属性设置外，datepicker()方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的 this 值等于对话框内容的 div 对象，不是整个对话框的 div。

## datepicker 事件选项

| 事件名 | 说明 |
| --- | --- |
| beforeShow | 日历显示之前会被调用。 |
| beforeShowDay | beforeShowDay(date)方法在显示日历中的每个日期时会被调用(date 参数是一个 Date 类对象)。该方法必须返回一个数组来指定每个日期的信息：<br>1.该日期是否可以被选择（数组的第一项，为 true 或 false）<br>2.该日期单元格上使用的 CSS 类<br>3.该日期单元格上显示的字符串提示信息。 |
| onChangeMonthYear | onChangeMonthYear(year, month,inst)方法在日历中显示的月份或年份改变时会被调用。或者 changeMonth 或 changeYear 为 true 时，下拉改变时也会触发。Year 当前的年，month 当年的月，inst 是一个对象，可以调用一些属性获取值。 |
| onClose | onClose(dateText, inst)方法在日历被关闭的时候调用。dateText 是当时选中的日期字符串，inst 是一个对象，可以调用一些属性获取值。 |

| | |
|---|---|
| onSelect | onSelect(dateText, inst)方法在选择日历的日期时被调用。dateText 是当时选中的日期字符串，inst 是一个对象，可以调用一些属性获取值。 |

```
//示例代码
$('#date').datepicker({
        beforeShow : function () {
        alert('日历显示之前触发！');
    },
    beforeShowDay : function (date) {
        if (date.getDate() == 1) {
            return [false,'a','不能选择'];
        } else {
            return [true];
        }
    },
    onChangeMonthYear : function (year,month,inst) {
        alert(year);
    },
    onClose : function (dateText,inst) {
        alert(dateText);
    },
    onSelect : function (dateText,inst) {
        alert(dateText);
    }
}) ;
```

注意：jQuery UI 只允许使用选项中定义的事件。目前还不可以试用 on()方法来管理。

## datepicker('action', param)方法

| 方法 | 返回值 | 说明 |
|---|---|---|
| datepicker('show') | jQuery 对象 | 显示日历 |
| datepicker('hide') | jQuery 对象 | 隐藏日历 |
| datepicker('getDate') | jQuery 对象 | 获取当前选定日历 |
| datepicker('setDate',date) | jQuery 对象 | 设置当前选定日历 |
| datepicker('destroy') | jQuery 对象 | 删除日历，直接阻断。 |
| datepicker('widget') | jQuery 对象 | 获取日历的 jQuery 对象 |
| datepicker('isDisabled') | jQuery 对象 | 获取日历是否禁用 |
| datepicker('refresh') | jQuery 对象 | 刷新一下日历 |

| | | |
|---|---|---|
| datepicker('option', param) | 一般值 | 获取 options 属性的值 |
| datepicker('option', param,value) | jQuery 对象 | 设置 options 属性的值 |

```
//显示日历
$('#date').datepicker('show');

//隐藏日历
$('#date').datepicker('hide');

//获取当前选定日期
alert($('#date').datepicker('getDate').getFullYear());

//设置当前选定日期
$('#date').datepicker('setDate', '2/15/2014');

//删除日历
$('#date').datepicker('destroy');

//获取日历的 jQuery 对象
$('#date').datepicker('widget');

//刷新日历
$('#date').datepicker('refresh');
//获取是否禁用日历
alert($('#date').datepicker('isDisabled'));

//获取属性的值
alert($('#date').datepicker('option', 'disabled'));

//设置属性的值
$('#date').datepicker('option', 'disabled', true);
```

# 验证插件

验证插件（validate.js），是一款验证常规表单数据合法性的插件。使用它，极大的解放了在表单上繁杂的验证过程，并且错误提示显示的完善也增加了用户体验。

## 使用 validate.js 插件

官网下载：http://bassistance.de/jquery-plugins/jquery-plugin-validation

最重要的文件是 validate.js，还有两个可选的辅助文件：additional-methods.js（控件 class 方式）和 message_zh.js（提示汉化）文件（实际使用，请使用 min 压缩版）。

**第一步：引入 validate.js**

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

**第二步：在 JS 文件中执行**

```
$('#reg').validate();
```

## 默认验证规则

Validate.js 的默认验证规则的写法有两种形式：

1.控件属性方式；

2.JS 键值对传参方式。

**默认规则列表**

| 规则名 | 说明 |
| --- | --- |
| required:true | 必须输入字段 |
| email:true | 必须输入正确格式的电子邮件 |
| url:true | 必须输入正确格式的网址 |
| date:true | 必须输入正确格式的日期（IE6 验证出错） |
| dateISO:true | 必须输入正确格式的日期(ISO（）只验证格式，不验证有效） |
| number:true | 必须输入合法的数字(负数，小数) |
| digits:true | 必须输入正整数 |
| creditcard:true | 必须输入合法的信用卡号，例如：5105105105105100 |
| equalTo:"#field" | 输入值必须和#field 相同 |
| minlength:5 | 输入长度最小是 5 的字符串(汉字算一个字符) |
| maxlength:10 | 输入长度最多是 10 的字符串(汉字算一个字符) |
| rangelength:[5,10] | 输入长度介于 5 和 10 之间的字符串") (汉字算一个字符) |

| | |
|---|---|
| range:[5,10] | 输入值必须介于 5 和 10 之间 |
| min:5 | 输入值不能小于 5 |
| max:10 | 输入值不能大于 10 |
| remote:"check.php" | 使用 ajax 方法调用 check.php 验证输入值 |

```
//使用控件方式验证 “必填和不得小于两位”
<input type="text" class="required" minlength="2" name="user" id="user" />
```

注意：默认规则里设置布尔值的，直接写到 class 里即可实现。如果是数字或数组区间，直接使用属性=值的方式即可。而对于错误提示是因为，可以引入中文汉化文件，或直接替换掉即可。

```
//使用 JS 对象键值对传参方式设置
$('#reg').validate({
    rules : {
            user : { //只有一个规则的话，
                    required : true, //直接 user : 'required',
                    minlength : 2,
            },
    },
    messages : {
            user : {
                    required : '帐号不得为空！',
                    minlength : '帐号不得小于 2 位！',
            },
    }
});
```

注意：由于第一种形式不能设置指定的错误提示信息。我们推荐使用第二种形式，第二种形式也避免的 HTML 污染。

```
//所有规则演示
$('#reg').validate({
    rules : {
            email : {
                    email : true,
            },
            url : {
                    url : true,
            },
            date : {
                    date : true,
            },
```

```
            dateIOS : {
                dateIOS : true,
            },
            number : {
                number : true,
            }
            digits : {
                digits : true,
            },
            creditcard : {
                creditcard : true,
            },
            min : {
                min : 5,
            },
            range : {
                range : [5, 10],
            },
            rangelength : {
                rangelength : [5,10],
            },
            notpass : {
                equalTo : '#pass', //这里的 To 是大写的 T
            }
    },
});
```

## validate()的方法和选项

除了默认的验证规则外，validate.js 还提供了大量的其他属性和方法供开发者使用。比如，调试属性，错误提示位置一系列比较有用的选项。

```
//jQuery.format 格式化错误提示
$('#reg').validate({
    rules : {
            user : {
                required : true,
                minlength : 5,
                rangelength : [5,10]
            },
    },
```

```
    messages : {
            user : {
                    required : '帐号不得为空！',
                    minlength : jQuery.format('帐号不得小于{0}位！'),
                    rangelength : jQuery.format('帐号限制在{0}-{1}位！'),
            },
    },
});

//开启调试模式禁止提交
$('#reg').validate({
            debug : true,
});

//设置调试模式为默认，可以禁止多个表单提交
$.validator.setDefaults({
    debug : true,
});

//使用其他方式代替默认提交
submitHandler : function (form) {
    //可以执行 ajax 提交，并且无须 debug:true 阻止提交了
},

//忽略某个字段验证
ignore : '#pass',

//设置错误提示的 class 名
errorClass : 'error_list',

//设置错误提示的标签
errorElement : 'p',

//统一包裹错误提示
errorLabelContainer : 'ol.error',
wrapper : 'li',

//设置成功后加载的 class
success : 'success',
```

```
//使用方法加载 class 并添加文本
success : function (label) {
    label.addClass('success').text('ok');
},

//高亮显示有错误的元素，变色式
highlight: function(element, errorClass) {
    $(element).fadeOut(function() {
            $(element).fadeIn()
    })
},

//高亮显示有错误的元素，变色式
highlight: function(element, errorClass) {
    $(element).css('border', '1px solid red');
},

//成功的元素移出错误高亮
unhighlight : function (element, errorClass) {
    $(element).css('border', '1px solid #ccc');
},

//表单提交时获取信息
invalidHandler : function (event, validator) {
    var errors = validator.numberOfInvalids();
    if (errors) {
            $('.myerror').html('您有' + errors + '个表单元素填写非法！');
    }
},

//获取错误提示句柄，不用提交及时获取值
showErrors : function (errorMap, errorList) {
    var errors = this.numberOfInvalids();
    if (errors) {
            $('.myerror').html('您有' + errors + '个表单元素填写非法！');
    } else {
            $('.myerror').hide();
    }
    this.defaultShowErrors(); //执行默认错误
},
```

```
//获取错误提示句柄，errorList
showErrors : function (errorMap, errorList) {
    alert(errorList[0].message); //得到错误信息
    alert(errorList[0].element); //当前错误的表单元素
},
```

# validate.js 其他功能

使用 remote:url，可以对表单进行 ajax 验证，默认会提交当前验证的值到远程地址。如果需要提交其他的值，可以使用 data 选项。

```
//使用 ajax 验证
rules : {
    user : {
            required : true,
            minlength : 2,
            remote : 'user.php',
    },
},
//user.php 内容
<?php
if ($_GET['user'] == 'bnbbs') {
    echo 'false';
} else {
    echo 'true';
}
?>
注意：远程地址只能输出'true'或'false'，不能输出其他值。
//同时传递多个值到远程端
pass : {
    required : true,
    minlength : 6,
    remote : {
            url : 'user.php',
            type : 'POST',
            dataType : 'json',
            data : {
                    user : function () {
                            return $('#user').val();
                    },
```

```
            },
        },
    },
    //user.php 内容
    <?php
    if ($_POST['user'] != 'bnbbs' || $_POST['pass'] != '123456') {
        echo 'false';
    } else {
        echo 'true';
    }
    ?>
```

validate.js 提供了一些事件触发的默认值，这些值呢，大部分建议是不用更改的。

```
//取消提交验证
onsubmit : false, //默认是 true
```

注意：设置为 false 会导致直接传统提交，不会实现验证功能，一般是用于 keyup/click/blur 验证提交。

```
//设置鼠标离开不触发验证
onfocusout : false, //默认为 true

//设置键盘按下弹起不触发验证
onkeyup : false, //默认为 true
```

注意：只要设置了，在测试的浏览器不管是 false 还是 true 都不触发了。

```
//设置点击 checkbox 和 radio 点击不触发验证
onclick : false, //默认为 true

//设置错误提示后，无法获取焦点
focusInvalid : false, //默认为 true

//提示错误时，隐藏错误提示，不能和 focusInvalid 一起用，冲突
focusCleanup : true, //默认为 false
```

如果表单元素设置了 title 值，且 messages 为默认，就会读取 title 值的错误信息，我们可以通过 ignoreTitle : true，设置为 true，屏蔽这一个功能。

```
ignoreTitle : true, //默认为 false

//判断表单所验证的元素是否全部有效
```

```
alert($('#reg').valid()); //全部有效返回 true
```

Validate.js 提供了可以单独验证每个表单元素的 rules 方法，不但提供了 add 增加验证，还提供了 remove 删除验证的功能。

```
//给 user 增加一个表单验证
$('#user').rules('add', {
    required : true,
    minlength : 2,
    messages : {
            required : '帐号不得为空！',
            minlength : jQuery.format('帐号不得小于{0}位！'),
    }
});

//删除 user 的所有验证规则
$('#user').rules('remove');

//删除 user 的指定验证规则
$('#user').rules('remove', 'minlength min max');

//添加自定义验证
$.validator.addMethod('code', function (value, element) {
            var tel = /^[0-9]{6}$/;
            return this.optional(element) || (tel.test(value));
}, '请正确填写您的邮政编码');

//调用自定义验证
rules : {
    code : {
            required : true,
            code : true,
    }
},
```

# 验证注册表单

本节课，将使用 validate.js 验证插件功能，完成表单注册验证的功能。

## html 部分

html 部分几部不需要更改太多，只要加个存放错误提示的列表标签即可。

```
<ol class="reg_error"></ol>
```

## css 部分

css 部分主要是成功后引入一张小图标，还有错误列表样式。

```
#reg p .star {
    color:maroon;
}
#reg p .success {
    display:inline-block;
    width:28px;
    background:url(../img/reg_succ.png) no-repeat;
}
#reg ol {
    margin:0;
    padding:0 0 0 20px;
    color:maroon;
}
#reg ol li {
    height:20px;
}
```

## jQuery 部分

jQuery 部分很常规，基本使用了 validate.js 的核心功能。

```
$('#reg').dialog({
    autoOpen : false,
    modal : true,resizable : false,
    width : 320,
    height : 340,
    buttons : {
            '提交' : function () {
                    $(this).submit();
            }
    },
}).buttonset().validate({
    submitHandler : function (form) {
```

```
        alert('验证完成，准备提交！');
},
showErrors : function (errorMap, errorList) {
        var errors = this.numberOfInvalids();
        if (errors > 0) {
                $('#reg').dialog('option', 'height', 20 * errors + 340);
        } else {
                $('#reg').dialog('option', 'height', 340);
        }
        this.defaultShowErrors();
},
highlight: function (element, errorClass) {
                $(element).css('border', '1px solid #630');
},
nhighlight : function (element, errorClass) {
        element).css('border', '1px solid #ccc');
        element).parent().find('span').html(' ').addClass('succ');
,
rrorLabelContainer : 'ol.reg_error',
        wrapper : 'li',
        rules : {
                user : {
                        required : true,
                        minlength : 2,
                },
                pass : {
                        required : true,
                        minlength : 6,
                },
                email : {
                        required : true,
                        email : true,
                },
                date : {
                        date : true,
                },
        },
        messages : {
                user : {
                        required : '帐号不得为空！',
```

```
                minlength : jQuery.format('帐号不得小于{0}位！'),
            },
            pass : {
                required : '密码不得为空！',
                minlength : jQuery.format('密码不得小于{0}位！'),
            },
            email : {
                required : '邮箱不得为空！',
                email : '请输入正确的邮箱格式！',
            },
            date : {
                date : '请输入正确的日期！',
            },
        },
});
```

# Ajax 表单插件

传统的表单提交，需要多次跳转页面，极大的消耗资源也缺乏良好的用户体验。而这款 form.js 表单的 Ajax 提交插件将解决这个问题。

## 核心方法

官方网站：http://malsup.com/jquery/form/

form.js 插件有两个核心方法：ajaxForm()和 ajaxSubmit()，它们集合了从控制表单元素到决定如何管理提交进行的功能。

```
//ajaxForm 提交方式
$('#reg').ajaxForm(function () {
    alert('提交成功！');
});
```

注意：使用 ajaxForm()方法，会直接实现 ajax 提交。自动阻止了默认行为，而它提交的默认页面是 form 控件的 action 属性的值。提交的方式是 method 属性的值。

```
//ajaxSubmit()提交方式
$('#reg').submit(function () {
        $(this).ajaxSubmit(function () {
            alert('提交成功！');
```

```
        });
        return false;
});
```

注意：ajaxForm()方法，是针对 form 直接提交的，所以阻止了默认行为。而 ajaxSubmit()方法，由于是针对 submit()方法的，所以需要手动阻止默认行为。而使用了 validate.js 验证插件，那么 ajaxSubmit()比较适合我们。

# option 参数

option 参数是一个以键值对传递的对象，可以通过这个对象，设置各种 Ajax 提交的功能。

```
$('#reg').submit(function () {
    $(this).ajaxSubmit({
        url : 'test.php', //设置提交的 url，可覆盖 action 属性
        target : '#box', //服务器返回的内容存放在#box 里
        type : 'POST', //GET,POST
        dataType : null, //xml,json,script，默认为 null
        clearForm : true, //成功提交后，清空表单
        resetForm : true, //成功提交后，重置表单
        data : { //增加额外的数据提交
            aaa : 'bbb',
            ccc : 'ddd'.
        },
        beforeSubmit : function (formData, jqForm, options) {
            alert(formData[0].name); //得到传递表单元素的 name
            alert(formData[0].value); //得到传递表单元素的 value
            alert(jqForm); //得到 form 的 jquery 对象
            alert(options); //得到目前 options 设置的属性
            alert('正在提交中！！！');
            return true;
        },
        success : function (responseText, statusText) {
            alert(responseText + statusText); //成功后回调
        },
        error : function (event, errorText, errorType) { //错误时调用
            alert(errorText + errorType);
        },
    });
    return false;
```

```
});
```

# 工具方法

form.js 除了提供两个核心方法之外，还提供了一些常用的工具方法。这些方法主要是在提交前或后对数据或表单进行处理的。

```
//表单序列化
alert($('#reg').formSerialize());

//序列化某一个字段
alert($('#reg #user').fieldSerialize());

//得到某个字段的 value 值
alert($('#reg #user').fieldValue());

//重置表单
$('#reg').resetForm()
```

# Ajax 提交表单

本节课，运用两大表单插件，完成数据表新增的工作。

## 一．创建数据库

创建一个数据库，名称为：zhiwen。表为：id、user、pass、email、sex、birthday、date。所需的 PHP 文件：config.php、add.php、is_user.php。

```
//config.php
<?php
header('Content-Type:text/html; charset=utf-8');
define('DB_HOST', 'localhost');
define('DB_USER', 'root');
define('DB_PWD', '123456');
define('DB_NAME', 'zhiwen');
$conn = @mysql_connect(DB_HOST, DB_USER, DB_PWD)
        or die('数据库链接失败：'.mysql_error());
@mysql_select_db(DB_NAME) or die('数据库错误：'.mysql_error());
@mysql_query('SET NAMES UTF8') or die('字符集错误：'.mysql_error());
?>

//add.php
<?php
require 'config.php';
$query = "INSERT INTO user (user, pass, email, sex, birthday, date)
VALUES ('{$_POST['user']}', sha1('{$_POST['pass']}'), '{$_POST['email']}',
   '{$_POST['sex']}', '{$_POST['birthday']}', NOW())";
mysql_query($query) or die('新增失败！'.mysql_error());
echo mysql_affected_rows();
mysql_close();
?>
//is_user.php
<?php
require 'config.php';
$query = mysql_query("SELECT user FROM user
        WHERE user='{$_POST['user']}'")or die('SQL 错误！');
if (mysql_fetch_array($query, MYSQL_ASSOC)) {
   echo 'false';
```

```
} else {
    echo 'true';
}
mysql_close();
?>
```

# 二．Loading 制作

在提交表单的时候，用于网络速度问题，可能会出现不同时间延迟。所以，为了更好的用户体验，在提交等待过程中，设置 loading 是非常有必要的。

```
//采用对话框式
$('#loading').dialog({
    modal : true,
    autoOpen : false,
    closeOnEscape : false, //按下 esc 无效
    resizable : false,
    draggable : false,
    width : 180,
    height: 50,
}).parent().parent().find('.ui-widget-header').hide(); //去掉 header 头

//CSS 部分
#loading {
    background:url(../img/loading.gif) no-repeat 20px center;
    line-height:25px;
    font-size:14px;
    font-weight:bold;text-indent:40px;
    color:#666;
}
```

# 三．Ajax 提交

最后，我们需要采用 form.js 插件对数据进行提交。而且在其他部分需要做一些修改。

```
submitHandler : function (form) {
    $(form).ajaxSubmit({
            url : 'add.php',
            type : 'POST',
            beforeSubmit : function (formData, jqForm, options) {
```

```
            $('#loading').dialog('open');
            $('#reg').dialog('widget').find('button').eq(1).button('disable');
        },
        success : function (responseText, statusText) {
            $('#reg').dialog('widget').find('button').eq(1).button('enable');
            if (responseText) {
                $('#loading').css('background', 'url(img/success.gif)
                    no-repeat 20pxcenter').html('数据提交成功...');
                setTimeout(function () {
                    $('#loading').dialog('close');
                    $('#loading').css('background', 'url(img/loading.gif)
                    no-repeat 20px center').html('数据交互中...');
                    $('#reg').dialog('close');
                    $('#reg').resetForm();
                    $('#reg span.star').html('*').removeClass('success');
                }, 1000);
            }
        },
    });
```

# cookie 插件

Cookie 是网站用来在客户端保存识别用户的一种小文件。一般来用库可以保存用户登录信息、购物数据信息等一系列微小信息。chrome://settings/cookies

## 一．使用 cookie 插件

```
//生成一个 cookie：
//查看 cookie：chrome://settings/cookies
$.cookie('user', 'htf');

//设置 cookie 参数
$.cookie('user', 'htf', {
    expires : 7, //过期时间，7 天后
    path : '/', //设置路径，上一层
    domain : 'www.baidu.com', //设置域名
```

```
    secure : true, //默认为 false，需要使用安全协议 https
});

//关闭编码/解码，默认为 false
$.cookie.raw = true;

//读取 cookie 数据
alert($.cookie('user'));

//读取所有 cookie 数据
alert($.cookie());
```
注意：读取所有的 cookie 是以对象键值对存放的，所以，也可以$.cookie().user 获取。

```
//删除 cookie
$.removeCookie('user');
```

## 二．注册直接登录

把 cookie 引入到知问前端中去。

```
//HTML 部分
<div class="header_member">
    <a href="javascript:void(0)" id="reg_a">注册</a>
    <a href="javascript:void(0)" id="member">用户</a>
    |
    <a href="javascript:void(0)" id="login_a">登录</a>
    <a href="javascript:void(0)" id="logout">退出</a>
</div>

//jQuery 部分
$('#member, #logout').hide();
if ($.cookie('user')) {
    $('#member, #logout').show();
    $('#reg_a, #login_a').hide();
} else {
    $('#member, #logout').hide();
    $('#reg_a, #login_a').show();
}
```

```
    $('#logout').click(function () {
        $.removeCookie('user');
        window.location.href = '/jquery/';
    });

    //登录成功之后的操作
    success : function (responseText, statusText) {
        $('#reg_a, #login_a').hide();
        $('#member, #logout').show();
        $('#member').html($.cookie('user'));
    }
```

# Ajax 登录

## 服务器端代码

```
//is_user.php
<?php
require 'config.php';
$query = mysql_query("SELECT user FROM user WHERE user='{$_POST['user']}'")
or die('SQL 错误！');
if (mysql_fetch_array($query, MYSQL_ASSOC)) {
        echo 'false';
} else {
        echo 'true';
}
mysql_close();
?>
```

注意：在部分环境下，要设置 UTF8 无 BOM，否则验证无法提醒。

```
//login.php
<?php
require 'config.php';
$_pass = sha1($_POST['login_pass']);
$query = mysql_query("SELECT user,pass FROM user WHERE
user='{$_POST['login_user']}' AND pass='{$_pass}'") or die('SQL 错误！');
if (mysql_fetch_array($query, MYSQL_ASSOC)) {
        echo 'true';
```

```
} else {
        echo 'false';
}
mysql_close();
?>
```

# jQuery 代码

```
//判断是否一周有效
if ($('#expires').is(':checked')) {
        $.cookie('user', $('#login_user').val(), {
                expires : 7,
        });
} else {
        $.cookie('user', $('#login_user').val());
}
```

# 选项卡 tab 组件

选项卡（tab），是一种能提供给用户在同一个页面切换不同内容的 UI。 尤其是在页面布局紧凑的页面上，提供了非常好的用户体验。

## 使用 tabs

使用 tabs 比较简单，但需要按照指定的规范即可。

```
//HTML 部分
<div id="tabs">
        <ul>
                <li><a href="#tabs1">tab1</a></li>
                <li><a href="#tabs2">tab2</a></li>
                <li><a href="#tabs3">tab3</a></li>
        </ul>
        <div id="tabs1">tab1-content</div>
        <div id="tabs2">tab2-content</div>
        <div id="tabs3">tab3-content</div>
</div>

//jQuery 部分
$('#tabs').tabs();
```

## 修改 tabs 样式

在弹出的 tabs 对话框中，在火狐浏览器中打开 Firebug 或者右击->查看元素。这样，我们可以看看 tabs 的样式，根据样式进行修改。我们为了和网站主题符合，对 tabs 的标题背景进行修改。

```
//无须修改 ui 里的 CSS，直接用 style.css 替代掉
.ui-widget-header {
        background:url(../img/ui_header_bg.png);
}
```

```
//去掉外边框
#tabs {
        border:none;
}

//内容区域修饰
#tabs1, #tabs2, #tabs3 {
        height:100px;
        padding:10px;
        border:1px solid #aaa;
        border-top:none;
        position:relative;
        top:-2px;
}
```

# tabs()方法的属性

选项卡方法有两种形式：

1.tabs(options)，options 是以对象键值对的形式传参，每个键值对表示一个选项；

2.tabs('action', param)，action 是操作选项卡方法的字符串，param 则是 options 的某个选项。

## tabs 属性

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| collapsible | false/布尔值 | 当设置为 true 是，允许选项卡折叠对应的内容。默认值为 false，不会关闭对应内容。 |
| disabled | 无/数组 | 使用数组来指定禁用哪个选项卡的索引，比如：[0,1]来禁用前两个选项卡。 |
| event | click/字符串 | 触发 tab 的事件类型，默认为 click。可以设置 mouseover 等其他鼠标事件。 |
| active | 数组和布尔值 | 如果是数组，初始化时默认显示哪个 tab，默认值为 0。如果是布尔值，那么默认是否折叠。条件必须是 collapsible 值为 true。 |
| heightStyle | content/字符串 | 默认为 content，即根据内容伸展高度。Auto 则自动根据最高的那个为基准，fill 则是填充一定的可用高度。 |
| show | false/布尔值 | 切换选项卡时，默认采用淡入效果。 |
| hide | false 布尔值 | 切换选项卡时，默认采用淡出效果。 |

```
//示例代码
$('#tabs').tabs({
        collapsible : true,
        disabled : [0],
        event : 'mouseover',
        active : false,
        heightStyle : 'content',hide : true,
        show : true,
});
```

**注意：设置 true 后，默认为淡入淡出，如果想使用别的特效，可以使用以下表格中的字符串参数。**

**show 和 hide 可选特效**

| 特效名称 | 说明 |
| --- | --- |
| blind | 对话框从顶部显示或消失 |
| bounce | 对话框断断续续地显示或消失，垂直运动 |
| clip | 对话框从中心垂直地显示或消失 |
| slide | 对话框从左边显示或消失 |
| drop | 对话框从左边显示或消失，有透明度变化 |
| fold | 对话框从左上角显示或消失 |
| highlight | 对话框显示或消失，伴随着透明度和背景色的变化 |
| puff | 对话框从中心开始缩放。显示时“收缩”，消失时“生长” |
| scale | 对话框从中心开始缩放。显示时“生长”，消失时“收缩” |
| pulsate | 对话框以闪烁形式显示或消失 |

## tabs()方法的事件

除了属性设置外，tabs()方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。

## tab 事件选项

| 事件名 | 说明 |
| --- | --- |
| create | 当创建一个选项卡时激活此事件。该方法有两个参数(event, ui)，ui 参数有两个子属性 tab 和 panel，得到当前活动卡和内容选项的对象。 |
| activate | 当切换一个活动卡时，启动此事件。该方法有两个参数(event, ui)，ui 参数有四个子属性 newTab、newPanel、oldTab，oldPanel。分别得到的时候新 tab 对象、新内容对象、旧 tab 对象和旧内容对象。 |
| beforeActivate | 当切换一个活动卡之前，启动此事件。该方法有两个参数(event, ui)，ui 参数有四个子属性 newTab、newPanel、oldTab，oldPanel。分别得到的时候新 tab 对象、新内容对象、旧 tab 对象和旧内容对象。 |
| load | 当 ajax 加载一个文档后激活此事件。该方法有两个参数(event, ui)，ui 参数有两个子属性 tab 和 panel，得到当前活动卡和内容选项的对象。 |
| beforeLoad | 当 ajax 加载一个文档前激活此事件。该方法有两个参数(event, ui)，ui 参数有四个子属性 tab 和 panel 以及 jqXHR 和 ajaxSettings，前两个得到当前活动卡和内容选项的对象，后两个是 ajax 操作对象。 |

```
//当选项卡创建时触发
$('#tabs').tabs({
        create : function (event, ui) {
                alert($(ui.tab.get()).html());
                alert($(ui.panel.get()).html());
        },
});

//当切换到一个活动卡时触发
$('#tabs').tabs({
        activate : function (event, ui) {
                alert($(ui.oldTab.get()).html());
                alert($(ui.oldPanel.get()).html());
                alert($(ui.newTab.get()).html());
```

```
            alert($(ui.newPanel.get()).html());
        },
});

//当切换到一个活动卡之前触发
$('#tabs').tabs({
        beforeActivate : function (event, ui) {
            alert($(ui.oldTab.get()).html());
            alert($(ui.oldPanel.get()).html());
            alert($(ui.newTab.get()).html());
            alert($(ui.newPanel.get()).html());
        },
});
```

在使用 load 和 beforeLoad 事件之前，我们先要了解一下 ajax 调用的基本方法。

```
//HTML 部分
<ul>
        <li><a href="tabs1.html">tab1</a></li>
        <li><a href="tabs2.html">tab2</a></li>
        <li><a href="tabs3.html">tab3</a></li>
</ul>
```

而 tabs1.html、tabs2.html 和 tabs3.html 只要书写即可，无须包含 。

比如：

tabs1-content 而这个时候，我们的 CSS 需要做一定的修改，只要将之前的 ID 换成如下即可：

```
#ui-tabs-1, #ui-tabs-2, #ui-tabs-3 {}

//ajax 加载后触发
$('#tabs').tabs({
        load : function (event, ui) {
            alert('ajax 加载后触发！');
        }
});
//ajax 加载前触发
$('#tabs').tabs({
        beforeLoad : function (event, ui) {
            ui.ajaxSettings.url = 'tabs2.html';
            ui.jqXHR.success(function (responseText) {
```

```
                alert(responseText);
            });
        }
});
```

## tabs('action', param)方法

| 方法 | 返回值 | 说明 |
| --- | --- | --- |
| tabs('disable') | jQuery 对象 | 禁用选项卡 |
| tabs('enable') | jQuery 对象 | 启用选项卡 |
| tabs('load') | jQuery 对象 | 通过 ajax 获取选项卡内容 |
| tabs('widget') | jQuery 对象 | 获取选项卡的 jQuery 对象 |
| tabs('destroy') | jQuery 对象 | 删除选项卡，直接阻断了 tabs。 |
| tabs('refresh') | jQuery 对象 | 更新选项卡，比如高度。 |
| tabs('option', param) | 一般值 | 获取 options 属性的值 |
| tabs('option', param, value) | jQuery 对象 | 设置 options 属性的值 |

```
//禁用选项卡
$('#tabs').tabs('disable'); //$('#tabs').tabs('disable', 0);

//启用选项卡
$('#tabs').tabs('enable'); //$('#tabs').tabs('enable', 0);

//获取选项卡 jQuery 对象
$('#tabs').tabs('widget');

//更新选项卡
$('#tabs').tabs('refresh');

//删除 tabs 选项卡
$('#tabs').tabs('destroy');

//重载指定选项卡内容
$('#button').click(function () {
        $('#tabs').tabs('load', 0);
});

//得到 tabs 的 options 值
alert($('#tabs').tabs('option', 'active'));
```

```
//设置 tabs 的 options 值
$('#tabs').tabs('option', 'active', 1);
```

## tabs 中使用 on()

在 tabs 的事件中，提供了使用 on()方法处理的事件方法。

**on()方法触发的选项卡事件**

| 事件名称 | 说明 |
| --- | --- |
| tabsload | Ajax 加载后触发 |
| tabsbeforeload | Ajax 加载前触发 |
| tabsactivate | 选项卡切换时触发 |
| tabsbeforeactivate | 选项卡切换前触发 |

```
//ajax 加载后触发
$('#tabs').on('tabsload', function () {
        alert('ajax 加载后触发！');
});

//ajax 加载前触发
$('#tabs').on('tabsbeforeload', function () {
        alert('ajax 加载前触发！');
});

//选项卡切换时触发
$('#tabs').on('tabsactivate', function () {
        alert('选项卡切换时触发！');
});

//选项卡切换前触发
$('#tabs').on('tabsbeforeactivate ', function () {
        alert('选项卡切换前触发！');
});
```

# 折叠菜单 UI

折叠菜单（accordion），和选项卡一样也是一种在同一个页面上切换不同内容的功能 UI。它

和选项卡的使用几乎没有什么太大区别，只是显示的效果有所差异罢了。

使用 accordion

使用 accordion 比较简单，但需要按照指定的规范即可。

```
//HTML 部分
<div id="accordion">
        <h1>菜单 1</h1>
        <div>内容 1</div>
        <h1>菜单 2</h1>
        <div>内容 2</div>
        <h1>菜单 3</h1>
        <div>内容 3</div>
</div>

//jQuery 部分
$('#accordion').accordion();
```

# 修改 accordion 样式

在显示的 accordion 折叠菜单中，在火狐浏览器中打开 Firebug 或者右击->查看元素。这样，我们可以看看 accordion 的样式，根据样式进行修改。我们为了和网站主题符合，对 accordion 的标题背景进行修改。

```
//无须修改 ui 里的 CSS，直接用 style.css 替代掉
.ui-widget-header {
        background:url(../img/ui_header_bg.png);
}
```

# accordion()方法的属性

选项卡方法有两种形式：

1.accordion(options)，options 是以对象键值对的形式传参，每个键值对表示一个选项；

2.accordion('action', param)，action 是操作选项卡方法的字符串，param 则是 options 的某个选项。

## accordion 常见属性

| 属性 | 默认值/类型 | 说明 |
| --- | --- | --- |
| Collapsible | false/布尔值 | 当设置为 true 是，允许菜单折叠对应的内容。默认值为 false，不会关闭对应内容。 |
| disabled | 无/布尔值 | 默认为 false，设置为 true 则禁用折叠菜单。 |
| event | click/字符串 | 触发 accordion 的事件类型，默认为 click。可以设置 mouseover 等其他鼠标事件。 |
| active | 数字和布尔值 | 如果是数字，初始化时默认显示哪个 tab，默认值为 0。如果是布尔值，那么默认是否折叠。条件 |
| heightStyle | content/字符串 | 默认为 auto，即自动根据最高的那个为基准必须，fill 则是填充一定的可用高度，content 则是根据内容伸展高度。collapsible 值为 true。 |
| header | h1/字符串 | 设置折叠菜单的标题标签。 |
| icons | 默认图标 | 设置想要的图标。 |

```
//示例代码
$('#accordion').accordion({
        collapsible : true,
        disabled : true,
        event : 'mouseover',
        active : 1,
        active : true,
        heightStyle : 'content',
        header : 'h3',
        icons: {
                "header": "ui-icon-plus",
                "activeHeader": "ui-icon-minus",
        },
});
```

# 三．accordion()方法的事件

除了属性设置外，accordion()方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。

## accordion 事件选项

| 事件名 | 说明 |
| --- | --- |
| create (event, ui) ， | 当创建一个折叠菜单时激活此事件。该方法有两个参数 ui 参数有两个子属性 header 和 panel，得到当前标题和内容选项的对象。 |
| activate | 当切换一个折叠菜单时，启动此事件。该方法有两个参数(event, ui)，ui 参数有四个子属性 newHeader、newPanel、oldHeader，oldPanel。分别得到的时候新 header 对象、新内容对象、旧 header 对象和旧内容对象。 |
| beforeActivate | 当切换一个折叠菜单之前，启动此事件。该方法有两个参数 (event, ui)，ui 参数 有四 个子属 性 newHeader、newPanel、oldHeader，oldPanel。分别得到的时候新 header 对象、新内容对象、旧 header 对象和旧内容对象。 |

```
//当折叠菜单创建时触发
$('#accordion').accordion({
	create : function (event, ui) {
		alert($(ui.header.get()).html());
		alert($(ui.panel.get()).html());
	},
});

//当切换到一个菜单时触发
$('#accordion').accordion({
	activate : function (event, ui) {
		alert($(ui.oldHeader.get()).html());
		alert($(ui.oldPanel.get()).html());
		alert($(ui.newHeader.get()).html());
		alert($(ui.newPanel.get()).html());
	},
});

//当切换到一个菜单之前触发
$('#accordion').accordion({
	beforeActivate : function (event, ui) {
		alert($(ui.oldHeader.get()).html());
```

```
            alert($(ui.oldPanel.get()).html());
            alert($(ui.newHeader.get()).html());
            alert($(ui.newPanel.get()).html());
        },
});
```

# accordion('action', param)方法

## accordion('action', param)方法

| 方法 | 返回值 | 说明 |
| --- | --- | --- |
| accordion('disable') | jQuery 对象 | 禁用折叠菜单 |
| accordion('enable') | jQuery 对象 | 启用折叠菜单 |
| accordion('widget') | jQuery 对象 | 获取折叠菜单的 jQuery 对象 |
| accordion('destroy') | jQuery 对象 | 删除折叠菜单 ，直接阻断了 accordion。 |
| accordion('destroy') | jQuery 对象 | 删折叠菜 单，直接阻断了 accordion。 |
| accordion('option', param) | 一般值 | 获取 options 属性的值 |
| accordion('option', param,value) | jQuery 对象 | 设置 options 属性的值 |

```
//示例代码
//禁用折叠菜单
$('#accordion').accordion('disable');

//启用折叠菜单
$('#accordion').accordion('enable');

//获取折叠菜单 jQuery 对象
$('#accordion').accordion('widget');

//更新折叠菜单
$('#accordion').accordion('refresh');

//删除 accordion 折叠菜单
$('#accordion').accordion('destroy');

//得到 accordion 的 options 值
alert($('#accordion').accordion('option', 'active'));
```

```
//设置 accordion 的 options 值
$('#accordion').accordion('option', 'active', 1);
```

# 五．accordion 中使用 on()

在 accordion 的事件中，提供了使用 on()方法处理的事件方法。

## on()方法触发的选项卡事件

| 事件名称 | 说明 |
| --- | --- |
| accordionactivate | 折叠菜单切换时触发 |
| accordionbeforeactivate | 折叠菜单切换前触发 |

```
//菜单切换时触发
$('#accordion').on('accordionactivate', function () {
        alert('菜单切换时触发！');
});

//菜单切换前触发
$('#accordion').on('accordionbeforeactivate ', function () {
        alert('菜单切换前触发！');
});
```