WEB 应用开发

轻轻松松学用 JavaScript 编程

软件技术文档

V0.1

Guo Yu
2004.02.19
12:52:42 +08'00'
I am the author
of this document

Last Change: 13-Feb-2004

SydongSun@hotmail.com

Author	Approved by
Name/Dept.:	Name/Dept.:
Date:	Date:
Signature:	Signature:

特别感谢 Green，我在 Green 的鼓励下完成本文的书写！

本文来自于作者（Sydongsun@hotmail.com）的一本书稿，该书面向基于 LOTUS DOMINO 的办公自动化（OA）软件开发。书名暂为《轻轻松松学用基于 Domino 的 WEB 开发》。该书稿中的其中有两个章节是关于 JavaScript 的一些简单介绍。本文基于该书稿的关于 JavaScript 的两个章节的内容，有所增加和删改，单列出来。便于其他学习 WEB 应用开发的初学 JavaScript 的参考。

由此可以看出，本文并不是一本集中讨论 JavaScript 的读物。只是一本入门向导的快餐式的读物，适合初学者。

本文档的标题为《轻轻松松学用 JavaScript 编程》，前面的“轻轻松松”有两个含义：第一，是直接继承《轻轻松松学用基于 Domino 的 WEB 开发》的名字来源。第二，本文力求简约，简单，篇幅短小，阅读起来容易；从我本身作为 JavaScript 的学习者的角度来写，我也是一个初学者，无法写就复杂的内容；本文完整的组件包括三个部分：本文档本身；本文档中涉及的 JavaScript 代码例子文件；本文档配套的视频教程（有声有色的读物，并不是本文档的内容的简单重复，我自己听起来也常常入迷。真的非常棒！）；如果你拿到本文的话，一定要有本文配套的例子代码文件，否则学习来是不轻松的。如果你想更加轻松，让你的学习成为一种享受的话，建议你向作者索取视频教程。

本人乐于将自己所学到的东西，和大家共享。可以开展我所擅长的内容的培训，比如：
基于 IBM (Rational) ClearCase 的软件配置管理培训；
基于 IBM (Lotus) Lotus Domino 的办公自动化软件开发培训；
基于 Oracle 的数据库管理员的培训；
基于 SAP 的 ERP 某些模块的培训；
面向开发人员的 TCP/IP 网络协议簇的培训课程（以 C 语言讲述）；
Solaris 系统管理的培训

作者主要采用网络会议系统进行培训。我们排除距离上的限制，我们可以清晰的交谈。你可以看到我的桌面操作情况。如果你共享你的桌面的话，我可以远距离的来检查作业完成情况，帮助解决你开发中遇到的问题。

目录表

目录表. 3
配图列表 5
列表 6
1 说明 7
1.1 主要的内容 7
1.2 学习目标 7
2 了解 JavaScript 浏览器上的程序语言 8
2.1 JavaScript 操作对象的简单介绍 -- 属性和方法 8
2.2 JavaScript 代码的加入 10
2.2.1 加入 JavaScript 代码的方式一 10
2.2.2 加入 JavaScript 代码的方式二. 12
2.2.3 加入 JavaScript 代码的方式三. 12
3 JavaScript 常用对象的例子 14
3.1 一个最常用情景的例子 14
3.2 JavaScript 文档对象模型图. 17
3.3 使用单选钮 (Radio) 和多选钮 (Checkbox) 的例子 18
3.4 JavaScript 中的字符串和日期对象. 21
3.4.1 字符串对象. 21
3.4.2 日期对象 22
3.5 桢结构和框架窗口 (Frame,IFrame) 24
3.5.1 了解链接的 Target 属性 25
3.5.2 由 JavaScript，在不同的桢（窗口）间访问对象 27
3.5.3 了解 IFrame 27
4 正则表达式和模式匹配 28
4.1 定义正则表达式 28
4.2 字符类 30
4.3 正则表达式的应用例子 30
5 可参考学习的，精美的代码例子 31
5.1 预载入图片，实现导航按钮的动态效果 31
5.2 显示对象的提示信息 34
5.3 一个精美的日历 37
5.4 一个下拉菜单 38
5.5 类似于资源管理器的树图 42
5.6 一个很好的编辑器 45

6 总结和作业 46
6.1 总结 46
6.2 作业 47

配图列表

Figure 1: 网页显示效果图 1 10
Figure 2: 验证用户输入的网页例子 14
Figure 3: JavaScript 文档对象模型图 17
Figure 4: 使用 Radio 和 CheckBox 的例子 18
Figure 5: 含有桢结构的网页文档 24
Figure 6: 一个包含左右两个桢的例子 25
Figure 7: 动态变换图片的按钮的网页 31
Figure 8: 显示对象的提示信息的网页 34
Figure 9: 精美的日历的例子 37
Figure 10: 一个下拉菜单的网页例子 38
Figure 11: Windows 操作系统下的资源管理器 42
Figure 12: 一个采用 Tree 图组织内容的网页 43
Figure 13: 所见即所得的编辑器的网页例子 45

列表

Table 1: Html 对象的常用事件列举. 9
Table 2: 字符串对象的常用方法例举 21
Table 3: 日期对象的常用方法例举 22
Table 4: 一些 Target 属性值 26

1 说明

1.1 主要的内容

- 简单介绍 JavaScript 语言，JavaScript 所处理的对象－－属性和方法；
- JavaScript 代码加入 HTML 文档中的方法；
- JavaScript 最常用情景的一个例子－－验证用户输入；
- JavaScript 的文档对象模型图；
- JavaScript 的字符串对象和日期对象；
- 窗口对象和框架窗口（FrameSet、IFrame）；
- 正则表达式的模式（Pattern）匹配－－验证用户输入；

- 可参考的 JavaScript 代码：预载入图片实现导航按钮的动态变化效果；
- 可参考的 JavaScript 代码：显示对象的提示信息；
- 可参考的 JavaScript 代码：精美的日历；
- 可参考的 JavaScript 代码：下拉菜单；
- 可参考的 JavaScript 代码：类似于资源管理器的树图；
- 可参考的 JavaScript 代码：一个所见所得的编辑器；

从上面这个大纲来看，本文档的内容明显的分为两个部分。前半部分，也就是一章的内容，主要简单介绍 JavaScript，是基础内容，必须要好好的理解。而后面一部分内容，则是精心选择的代码，借此开拓学习者的视野。其中精美的日历，下拉菜单，树图是来自国外的共享软件开发者－－一些 JavaScript 专家写就的。在本文中加以介绍性引用（版权属于原作者），让大家入门之后，如果想进一步深入 JavaScript 的开发的话，这是优美的，足够我们学习的代码。当然如果你不想深入下去。直接想修改使用它们，建议在财力许可和方便的话，请付给这些作品的原作者相应酬劳，尊重他们的劳动成果。就像你的劳动成果也希望得到别人的尊重一样。

本文的格式上，尽量用褐色字体来表示代码。

本文中的一些名字概念的说法和其他的书本并不一定一致，这是作者的习惯的说法。但是不会对于你理解本文的内容有障碍。

为了更好的阅读效果，作者建议你打印本文档。本文档不到 50 页，如果你缩印成 50%打印的话，是很小的一本册子，也许会增加你学习的信心。

1.2 学习目标

通过本文档的阅读，开发人员应该掌握下列内容：

1．用 JavaScript 来访问浏览器环境和 HTML 语言所构成的现成对象的方法；
2．熟悉 JavaScript 文档对象模型；在心中有个清楚的轮廓；
3．进一步学习 JavaScript 的能力获得；

2 了解 JavaScript 浏览器上的程序语言

为了提供给最终用户以使用方便、界面美观的用户感受，很多基于特定的客户端程序，都转向趋向于采用浏览器客户端的应用。而早期的静态的、缺乏交互的 HTML 页面文档适应不了这种要求。JavaScript 是一种基于对象的脚本编程语言，是“浏览器”上的程序语言。当 Web 服务器输出内容（包括 JavaScript 的程序代码）到浏览器时，此时，**JavaScript 可以操纵浏览器上的一切内容**，**在浏览器上**提供用户交互，界面美化，增加 Web 界面的“智能性”。

我们知道：JavaScript 是一种脚本语言，是同 HTML 文档一起使用的，WEB 服务器产生输出 HTML 内容的同时，也输出一些 JavaScript 程序代码到浏览器客户端。JavaScript 由浏览器解释执行，不同的浏览器，或者是同一浏览器的不同版本，对 JavaScript 的支持会有一些差异。本书不讨论这些问题，并假定用户采用 IE5.0 以上版本的浏览器。JavaScript 是一种基于对象的语言，通过 JavaScript 代码来操作对象－－访问或者设置对象的属性，编写对象的特定的事件（方法）代码。

2.1 JavaScript 操作对象的简单介绍 -- 属性和方法

JavaScript 中的对象是由属性(properties)和方法(methods)两个基本的元素的构成的。前者是对象在实施其所需要行为的过程中，实现信息的装载单位，从而与变量相关联；后者是指对象能够按照设计者的意图而被执行，从而与特定的函数相联。

对象从哪里来？JavaScript 可以操作的对象有下列三个来源：

- 浏览器环境和 HTML 标签语句所构成的现成对象（链接、图像、插件、HTML 表单元素、浏览器细节等）；
- 通过 JavaScript 的内置类所创建的对象，比如 Data（日期）和 Number(数值)；
- 通过 JavaScript 编程，用户自己创建的对象；

本章的内容将涵盖以上三种情形的对象，一般来说，我们需要熟悉 JavaScript 的前两种的访问对象的方法。那些精通 JavaScript 程序编写的设计人员，他们常通过 JavaScript 程序，实现复杂的类，然后由类来创建对象。比如漂亮的日历、用于网站内容导航的下拉菜单、树状图，等等。我们只是 Domino 程序设计人员，除非我们能够编写比他们更好的 JavaScript 类，一般我们可以直接借用那些精通 JavaScript 编程的设计者的成果。Internet 的目的之一是满足资源和信息的共享，毫无疑问，这也包括能够让我们共享到那些精通 JavaScript 的编程者的优秀成果。

一般的对象有哪些方法（事件）呢？对于 HTML 标签语句所构成的现成的对象，我们列举一些用于说明。

在本书的第一部分，我们提及到：万事万物皆对象，日常生活中的对象具有各自的特性和行为。在面向对象的程序设计中，对应为属性和方法。在 JavaScript 的对象中，有很多以 on 开头的方法，比如 onLoad、onClick、onMouseOver、onMouseOut 等等，代表特定的事件。很多对象都具有以上方法。比如 onClick 事件，当鼠标点击某个对象时，这个对象可以是一个按钮、或者是表格的一行、一个图片、一个链接等。通过 onClick 所关联的 JavaScript 程序代码，做各种处理。

onMouseOver 和 onMouseOut 分别是指当鼠标移到和移出某个对象的区域时，所触发的事件，通过它们联接的 JavaScript 代码做各种处理。

我们来看下列 HTML 语句：

```
<INPUT class=btnmenuview onmouseover="this.className='btnmenuviewmouseover'"
onmouseout="this.className='btnmenuview'"  onClick=javascript:submitCreate();
name=btnHtmNew    title=新建申请表单 type=button  value=新建>
```

以上是一个“新建”按钮对象，从上面 HTML 标签语句，可以看出该按钮对象的属性有 Class、Name、Value。该按钮的事件代码有 onMouserOver 和 onMouseOut 以及 onClick。其中前两个事件（鼠标移到和移开按钮时）的作用是：变更该对象的 Class 属性值，在该页面的所引用的 CSS 文件中，分别指定了 btnmenuview 和 btnmenuviewmouseover 的外观风格，从而是当鼠标移动到按钮后，该按钮的外观会发生变化的动态效果。OnClick 事件是当用户点击按钮时，触发的 JavaScript 代码。这里将这些代码放置在 submitCreate()函数当中。

再来看看 onLoad 事件，当用户把 Web 页导入到浏览器之后，与该页（或文档）相关联的 onLoad 事件处理程序被执行；下面是调用该事件的例子：

```
<body onLoad="window.alert('页面内容完成导入到浏览器')">
```

onLoad 事件的应用较多，它的应用情景为：当网页到入到浏览器后，需要完成一些自动处理过程，这些处理过程（比如执行运算，或者显示信息）的 JavaScript 代码可以放在<body>标签所指的对象的 onLoad 事件所关联的函数中。

下面表格中列出常用的一些事件、这些事件的触发原因以及常常使用这些事件的对象例举：

Table 1: Html 对象的常用事件列举

事件名称	触发时间	对象例举
OnBlur	对象失去输入焦点	窗口和所有的表单对象
OnChange	用户改变对象的值	文本框、文本区域、选择列表等
OnClick	用户鼠标点击	链接、按钮、单选钮、多选钮等
OnFocus	获得输入焦点时	窗口和所有的表单对象
OnKeyDown	用户按下一个键时	表单对象，比如输入框、文本区域等
OnKeyUp	用户释放一个键时	表单对象，比如输入框、文本区域等
OnMouseDown	用户按下鼠标时	文档，按钮、链接、图像、表格等
OnMouseOut	鼠标从某个对象移开时	文档，按钮、链接、图像、表格等
OnMouseOver	鼠标移到某个对象	文档，按钮、链接、图像、表格等
OnReset	表单复位时	表单
OnResize	窗口尺寸变化时	窗口
OnSubmit	表单提交时	表单

2.2 JavaScript 代码的加入

从上一小节内容，我们已经对 JavaScript 中的对象、对象的属性方法和方法有了初步的了解，我们进一步了解下一个问题：JavaScript 代码应该放在 Web 页的什么位置呢？

JavaScript 的脚本包括在 HTML 中，它成为 HTML 文档的一部分。与 HTML 标识相结合，构成动态的、能够交互的网页。

JavaScript 的代码的加入，通过在 Web 页中直接写入：

```
<Script Language ="JavaScript">
    //JavaScript 语言代码；
      //JavaScript 语言代码；
      //…
</Script>
```

注释：

- 通过标识<Script> </Script>指明其中包含的是 Script 脚本代码；
- 通过 Language ="JavaScript"说明标识中使用的语言，这里是 JavaScript 语言；

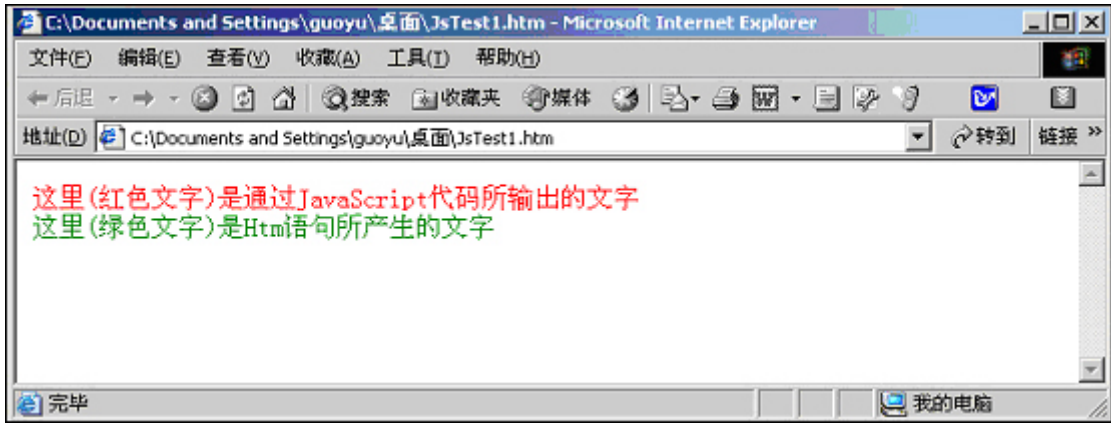
2.2.1 加入 JavaScript 代码的方式一

下面是一个 Htm 页面 JsTest1.htm 的源文件：

```
<HTML>
<Head>
<Script Language ="JavaScript">
    document.write("<font color=red>这里(红色文字)是通过 JavaScript 代码所输出的文字</font>");
</Script>
</Head>
<body>
<br>
<font color=green>这里(绿色文字)是 Htm 语句所产生的文字</font>
</body>
</HTML>
```

该页面显示的结果为：

Figure 1: 网页显示效果图 1



注释：

- document. write()是文档对象的输出函数，其功能是将括号中的字符或变量值输出到 Web 页面文档中；
- 可将<Script></Script>标识放入<Head>...</Head>或<Body>...</Body>之间。将 JavaScript 标识放置<Head>...</Head>在头部之间，使之在页面文档主体和其余部分代码之前装载。尤其是一些函数的代码，建议读者将这些代码放在<Head>... </Head>在头部之间。
- 也可以将 JavaScript 标识放置在<Body>... </Body>主体之间以实现某些部分动态地创建文档。

2.2.2 加入 JavaScript 代码的方式二

另一种加入 JavaScript 代码的方式，是在 HTML 标签所定义的对象当中，直接写某些事件的非常简短代码，比如像上一小节的列举的“新建”按钮的事件代码。如果这些事件的代码较为复杂的话，建议在<head></head>的<Script>..</Script>中间编写成一个函数，在按钮的标签中通过

onclick=javascript:函数名称()

来调用。

JavaScript 函数的写法是在<Script></Script>之间编写

```
function 函数名称(){
        //具体的处理过程
}
```

在这里提醒一下：JavaScript 语句的写法是区分大小写的，比如函数 submitCreate()和 SubmitCreate()并不一样。在 JavaScript 程序语句的 IF 语句应该为小写 if，如果不小心写为 If 是错误的。初学 JavaScript，要注意防止发生这种错误。

2.2.3 加入 JavaScript 代码的方式三

还有一种加入 JavaScript 代码方式是－－使用库函数：在 Web 页中加入类似于

<script src="指定的 Js 文件位置"></script>

来引入 JavaScript 库。

程序员谈到库（Library），并不是指可以借书的图书库（在程序员眼中，库就是一堆函数，有时是免费使用，有时候需要购买，当然也可以自己编写）。我们可以引用或在自己的代码中调用这些库函数。

要实现库，需要以.js 作为文件扩展名的文件来保存，在随书光盘的“JavaScript”目录下的“下拉菜单”目录下的 Index.htm 页面中有

```
<script language="JavaScript" src="menu_data.js"></script>
<script language="JavaScript" src="menu_script.js"></script>
```

来引入下拉菜单的数据文件库和下拉菜单的 Class 库文件。

把一些 JavaScript 代码（尤其是用户自己编写的对象的类文件）组织成可以反复使用的库，具有下列好处：

- 减少错误，减少 Web 页面的内容。JavaScript 库经过严格测试后，可以放心的反复调用，相对于通过拷贝和粘贴把 JavaScript 函数插入到每个想要调用它的 HTML 文件而言。同时也使 HTML 文件看起来清楚易读。
- 减少网络流量，提高响应速度。当一个 JavaScript 库的第一次下载到内存，无论多少页引用该库，浏览器都能访问它。不需要再次下载。

要决提醒:

考虑 JavaScript 脚本的位置，要注意下列两点：

- Web 内容在浏览器中是从上到下的顺序解释的。放在 HTML 的<head></head>之间脚本比插入 Web 的<body></body>的脚本先处理。比较好的做法是将包含所有预定义函数的脚本放在 Web 的<head></head>之间。这样，浏览器在前面遇到这些函数，确保 Web 的<body></body>中的语句能够识别这些函数。同样的道理，在一些网页下载到浏览器中，就会执行的脚本（比如 Web 页的 onload 事件关联的脚本代码），如果这些脚本要访问 HTML 标签所定义的对象，那么要确保这些对象先于脚本执行，否则会出现“对象不存在”的错误。建议设置 IE 的浏览器的高级属性中启用脚本调试，可以发现错误存在的地方。
- 应用外部脚本库<script language="JavaScript" src="menu_data.js"></script>，浏览器会在该 HTML 文件所在的目录下查找 menu_data.js 文件，如果把 js 文件存放在别的目录中，则 RSC 属性值必须反映出那个目录，也就是必须确保该 HTML 文件能够找到 js 文件。

3 JavaScript 常用对象的例子

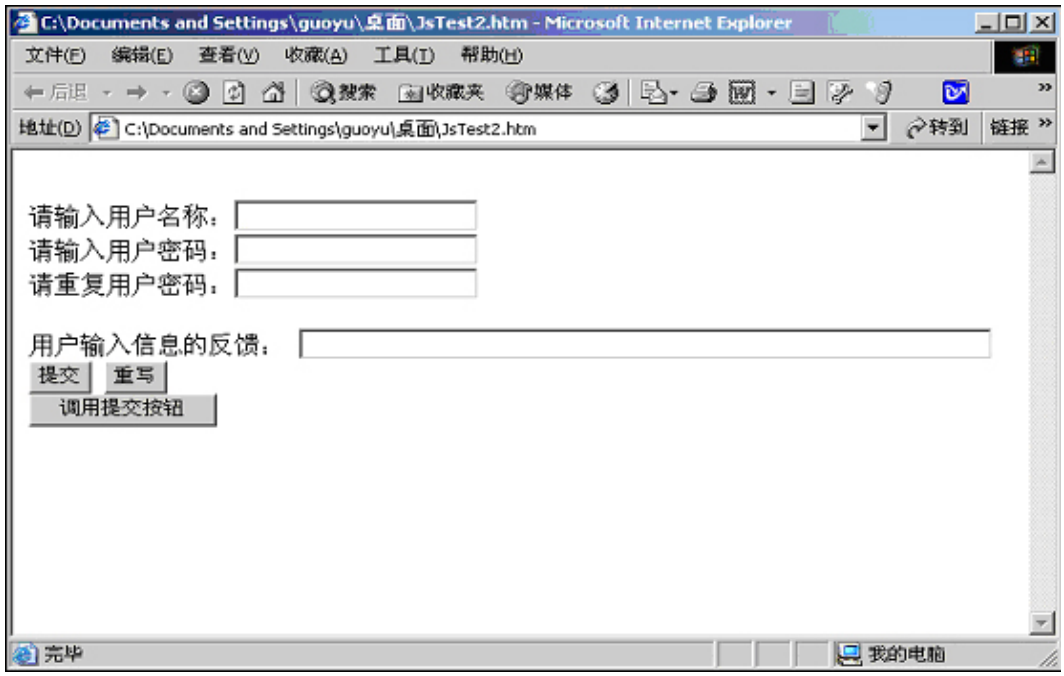
学习 JavaScript 比较好的入门方法是去学习参考别人的例子，而不必去寻找一本复杂 JavaScript 语法大全之类的书。只需要我们经常上网冲浪，发现别人的用法不错，刚好我们自己也需要那样做，那么就仿照他们的做法。下面给出 JavaScript 最经常使用的一个场景的简单例子。

3.1 一个最常用情景的例子

如下图 2：该网页通过 JavaScript 实现下列功能：

- 验证用户输入的功能，用户名称，用户密码，重复用户密码三项内容不能为空，同时两次密码录入应该一致。
- 如果用户的输入正确，那么在“用户输入信息的反馈”中显示一段反映用户录入的文本；

Figure 2: 验证用户输入的网页例子



以上页面的 HTML 代码为：

```
<HTML><Head>
<Script Language ="JavaScript">
function submitClick(){
        var Userid = document.forms[0].UserId.value;
        if(Userid==""){alert("用户名称不能为空");return false;}
        var UserPwd = document.forms[0].UserPwd.value;
        if(UserPwd==""){alert("用户密码不能为空");return false;}
        var reUserPwd = document.forms[0].reUserPwd.value;
        if(reUserPwd==""){alert("请重复密码输入");return false;}

        if (UserPwd!=reUserPwd){
                alert("两次输入密码不一致！请重新输入");
                document.forms[0].UserPwd.value ="";
                document.forms[0].reUserPwd.value ="";
                document.forms[0].UserPwd.focus();
                return false;
        }

        var strMsg ="";
        strMsg = "你的用户名称为：" + Userid + "; 你的密码为：" + reUserPwd +";";
        document.forms[0].DispInfo.value = strMsg
        alert(strMsg);
}

function ResetClick(){
        document.forms[0].reset();
}

function CallSubmitClick(){
        document.forms[0].Submit.click();
}

</Script></Head>
<body>
<form name="InputForm" method="post" action="">
<br>请输入用户名称：<input type="text" name="UserId"  id="UserIdid">
<br>请输入用户密码：<input type="password" name="UserPwd">
<br>请重复用户密码：<input type="password" name="reUserPwd">
<br>
<br>用户输入信息的反馈： <input type="text" name="DispInfo" size="60" readonly>
<br>
   <input type="button" name="BtnSubmit" value="提交" onclick=javascript:submitClick()>
   <input type="button" name="BtnReset" value="重写" onclick=javascript:ResetClick()>
<br>
   <input type="button" name="BtnCallSubmit" value="调用提交按钮" onclick=CallSubmitClick()>
</form>
</body>
```

详细分析：

◆ 分析代码行 var Userid = document.forms[0].UserId.value;

该语句的功能是将 Web 页面上的名称为 UserId 的 Text 对象的值赋给一个变量 Userid。JavaScirpt 访问 HTML 上标签所定义的对象，最常常采用的是一种**层层限定的逐步收缩法**。大家比较熟悉传统的邮件通信，如果要让邮件到达某个人，你首先要确定首先是那个国家（比如中国），然后是某个省（比如浙江），然后是某个城市（比如杭州），某个区（比如西湖区），某个大街（比如文三路），最后交代门牌号码（比如 158 号），然后就是那个人。

同传统的邮件通信类似，JavaScript 访问 Web 页面 HTML 标签语语句所构成的对象，也是**层层限定的逐步缩法**。在 JavaScript 的文档对象模型(DOM)中，窗口（Window）是对象模型的顶端对象，通常来说窗口就是你的浏览器。HTML 页面文档是在浏览器的窗口中显示的。目前我们假设页面不包含帧结构，在以后内容有专门的一个小节来讲述有关帧结构的问题。这里假设浏览器窗口中只显示一个 Web 页的情景。

浏览器的窗口（Window）有它的属性，比如它显示的页面，窗口（Window）底部的状态条上的文字等等；它也有方法，比如打开和关闭。通常来说，因为窗口在 JavaScript 的文档对象模型(DOM)对象层次的顶层，JavaScript 就假设 Window 已经存在了，你不必去在 JavaScript 程序中刻意写上它，也就是说“window.location”和“location”的作用是相同的。

窗口里是 Web 页面，它的对象层次从文档（document）开始。可以用 Window.document 来引用它，或者就是简单的 document，这同我们在国内邮件通信时，地址一般都不写“中国”。每个窗口只有一个文档（document）的时候。

一般情况下，有收集用户输入信息的文档（document）包含至少一个表单（form），但是可以包含多个。可以通过 document.forms[0]来访问第一个表单。当然表单一般都有名称（Name）属性，也可以通过表单的名称来访问，比如：上面的一句 JavaScript 也可以写成 var Userid = document.InputForm.UserId.value;

在 Microsoft 的 IE 浏览器环境下，可以不指明表单，还是直接用 all 替代（作者建议尽量不要采用这种办法），如：

var Userid = document.all.UserId.value;

表单中当中会包含很多 Input 对象，比如单行文本输入框（类型为 Text）、文本区域（类型为 TextArea）、普通按钮（类型为 Button）、提交按钮（类型为 Submit）、重置按钮（类型为 Reset）、选择框（类型为 Select）等等。

要访问例子中的用户名称录入的文本框 UserId 的 value 属性，可以通过

document.InputForm.UserId.value;

如果要将鼠标焦点停留该文本输入框中，可以通过该对象的 focus 方法：

document.InputForm.UserId.focus();

比如图 2-2 中“调用提交按钮”的 onclick 事件所关联的函数的语句为：

document.forms[0]. BtnSubmit.click();

含义是：访问到 document（文档）下的 forms[0]（表单）下的 BtnSubmit 按钮，调用该按钮的 Click 事件。

◆ 分析代码 document.forms[0].reset();

关于表单中的 Submit 和 Reset 类型的按钮：

HTML 的表单有个 Action 属性，该属性的值为某个页面的地址，当表单提交后，表单当中的用户输入内容将发送给 Action 所指定的页面，该页面做相应的处理，比如获得用户的输入，存入到数据库系统中去。图 2 的 Web 页面中没有指定 Action 的值。

对于表单对象来说，有下列两个方法 submit()和 reset()方法。前者对应表单的提交，后者对应表单内容的复位（初始状态），以便重新录入。图 2 中的“重写”按钮所关联的函数的代码

```
document.forms[0].reset();
```

就是调用表单的 Reset()方法；

表单中的 Input 对象有两种特别的按钮对象：类型为 Submit 的按钮对象和类型为 Reset 的按钮对象，点击这种按钮，它的动作就是触发表单的 submit()事件或者 reset()事件。在图 2 的例子中，我们采用类型为 Button 的普通按钮对象来完成这种功能。比如图 2 的“重写”按钮如果要用 Reset 类型的按钮替代的话，只需要直接

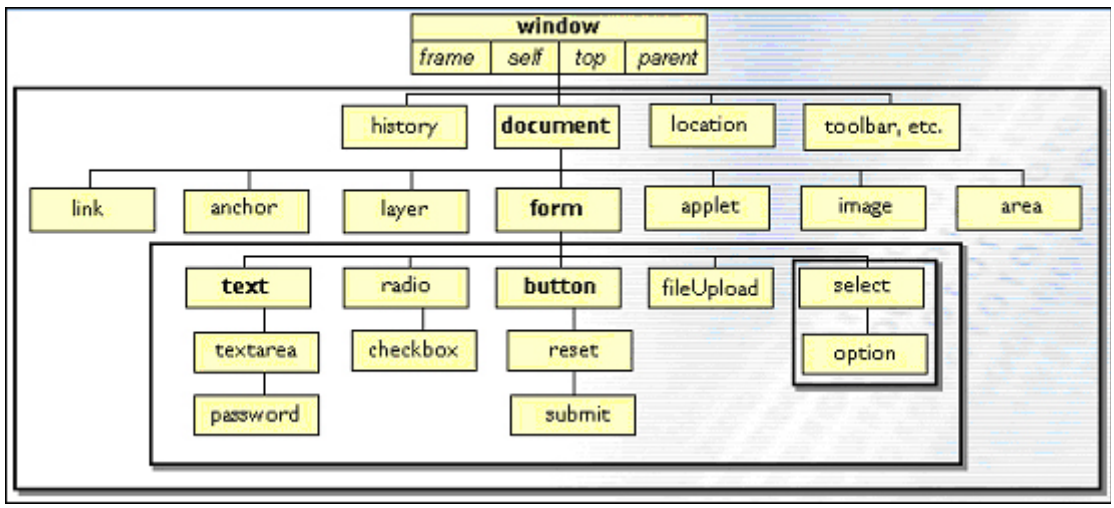
```
<input type="Reset" name="Reset" value="重写" >
```

不需要写它的 Onclick 事件代码函数，因为这种按钮的动作默认就是表单 reset()事件。

3.2 JavaScript 文档对象模型图

从上一个小节中的例子，我们已经得到了 JavaScript 访问 HTML 标签所定义的对象的方法。即**层层限定的逐步搜索法**。现在我们来看如下图：

Figure 3: JavaScript 文档对象模型图



通过这个图层次结构，我们可以进一步熟悉**层层限定的逐步搜索法**，请读者在这个图上花费几分钟的时间，在以后的一些内容中，我们也是围绕这个图，来讲述这个“图片背后的故事”。

顺便提一下，尽管我们常常采用**逐步收缩**的方法来访问 HTML 标签所定义的对象。有时候我们也可以才采用下列方法：

通过对象的 ID 或者对象的名称来获得该对象。比如：在图 2 中的用户名称的输入框 Text 的名称为 UserId，它的 ID 为 txtUserId，那么通过

```
var UserId =document.getElementById("txtUserId").value;
```

或者

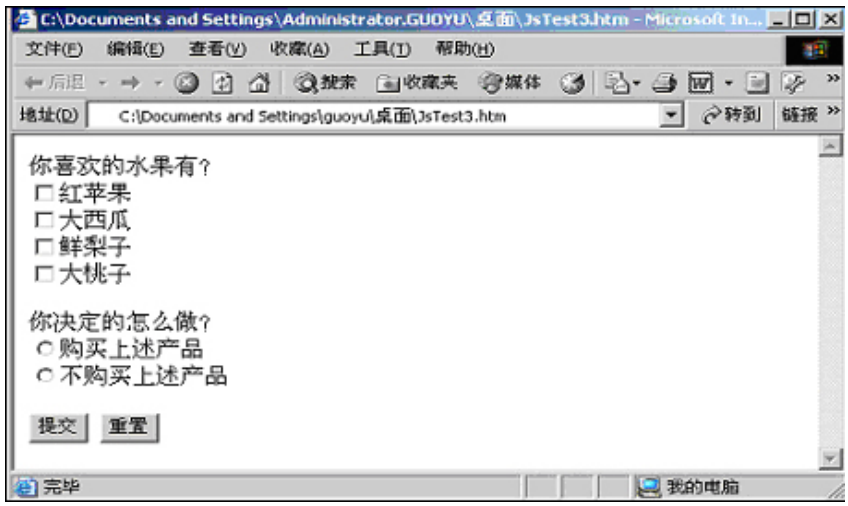
```
var UserId =document.getElementByName("UserId ").value
```

来取得该对象的 Value 的值。

3.3 使用单选钮（Radio）和多选钮（Checkbox）的例子

为了进一步加深对图 3 **JavaScript 文档对象模型图**的印象，我们列举一个网页中使用单选钮（Radio）和多选钮的（Checkbox）的例子。该 Web 页的程序清单在随书光盘的 JavaScript 目录下的 JsTest3.htm 中。该 Web 页显示的功能是：通过按钮来检测是否对单选钮（多选按钮）做了选择，并取得选择的单选钮（多选按钮）的值，如图：

Figure 4: 使用 Radio 和 CheckBox 的例子



该 Web 页（随书光盘 JavaScript 目录下的 JsTest3.htm）的源文件为：

```
<HTML><Head>
<Script Language ="JavaScript">
function submitClick(){
    var selValue = Select_check("cbx");
    if(selValue == 0){
            alert("你没有做出你喜爱的水果的任何选择，请作出选择！");
            return false;
    }else{
            var selActionValue = Select_check("radio1");
            if(selActionValue == 0){
                    alert("请你做出购买还是不够买的选择！");
                    return false;
            }else{
                    strMsg = "你喜欢的水果有" + selValue;
                    strMsg = strMsg + "，并且你决定了：" + selActionValue;
                    alert(strMsg);
            }
    }

}

function Select_check(objName){
    var obj=document.getElementsByName(objName);
    var selArray = new Array();
    for(var i=0;i<obj.length;i++){
            if(obj[i].checked){
                    selArray[i]=obj[i].value;
            }
    }
    return selArray;
}
</script>

</Script></Head><body>
<form name="InputForm" method="post" action="">
你喜欢的水果有？<br>
  <input type=checkbox name=cbx value=苹果>红苹果<br>
  <input type=checkbox name=cbx value=西瓜>大西瓜<br>
  <input type=checkbox name=cbx value=梨子>鲜梨子<br>
  <input type=checkbox name=cbx value=桃子>大桃子<br><br>

你决定的怎么做？<br>
  <input type=radio name=radio1 value="Buy">购买上述产品<br>
  <input type=radio name=radio1 value="Nobuy">不购买上述产品<br><br>

    <input type="button" name="Submit" value="提交" onclick=javascript:submitClick()>
    <input type="Reset" name="Reset" value="重置">
</form></body>
    </HTML>
```

简单注释：

JavaScript 脚本中函数 Select_check(objName)带有一个参数 objName，通过语句：

var obj=document.getElementsByName(objName);

来访问 Document 中的对象。

对于相关成组单选钮和多选钮，在 Web 上总是以同名的一组对象出现，在函数中，遍历对象组的每个元素，如果该元素的是否 Checked，如果是，则把该元素的值存入到数组 selArray 中，最后，函数返回值为数组 selArray。

要决提醒：

到此为止，我们已经遇到表单（Form）和文档（Document）的多重含义，我们不能混淆，要自然的理解作者在不同的情景下提到他们所指的含义:

- 在 Domino 系统中，设计元素对象表单（Form），用户通过表单来产生文档，并且通过表单来显示文档（在 Domino 的 Web 设计中，表单的作用有些扩展，有些表单只用来作内容的显示，比如显示嵌入视图的表单）。在 HTML 中也有表单对象（Form），HTML 表单的作用是用来收集用户录入，它会包含一些文本录入框(Text)，文本区域（TextArea）,按钮（Button），选择框（Select），复选框（Checkbox），单选框（Radio）等等。这是两种差别很大的不同的对象概念。尽管如此，他们也存在一些联系，在 Web 方式打开 Domino 系统的表单，或者打开文档处于编辑状态，Domino 系统将表单后者文档“翻译”成 HTML 页面，将自动的产生一个 HTML 的表单对象，而 Domino 系统的表单的不同类型的域将相应的转发为 HTML 表单对象中的不同 Input 表单对象，比如可编辑的文本域将“翻译”成一个文本录入框（Text）对象，而 RTF 可编辑域将“翻译”成文本区域（Textarea）。而这些 HTML 标签所定义的对象是 JavaScript 可以直接操作的对象。
- 在 Domino 系统中，文档（Document）对象是指 Domino 文本数据库系统所存放信息的单元。而在 JavaScript 的对象模型中，也有 Document 对象的概念，JavaScript 文档对象代表 Web 页中的内容。

对于图 3 JavaScript 对象模型的学习，我们通过两个例子有了对他们的初步了解。文档对象（DOM）模型描述了 JavaScript 用到的有关文档和浏览器的全部详细资料，包括那些对象具有什么样的属性和方法。

- 熟悉这些对象的比较好的办法就是学习别人的例子，本书当中包含了 Domino 的 Web 设计所要应用的大多数对象的例子。通读本书，基本上可以达到：JavaScript 不会成为在基于 Domino 的 Web 设计的一个障碍。
- 当然，本书并不是专门讨论 JavaScript 的书籍。深入学习 JavaScript 的最好方法就是多上网冲浪，看看别人的源代码。在 BBS 上同那些精通 JavaScript 的设计者进行求助和讨论。推荐两个网址：

  中国软件（www.csdn.net）的讨论区便于我们相互交流。

  MSDN 上的参考资源：

  http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dHTML/reference/objects.asp

  也可以多访问一些国外 JavaScript 开发专家的网站，研究学习他们的优秀的代码。

3.4 JavaScript 中的字符串和日期对象

3.4.1 字符串对象

像其他编程一样，字符串对象是 JavaScript 语句中经常要处理的对象，在这里我们来了解字符串对象的常见属性和方法。创建字符串对象，一般可以通过：

```
var strTemp =”This is a string”;
```

该句得到一个值为 This is a string 的字符串变量 strTemp。在 JavaScript 中，经常需要获得 HTML 标签所定义的 Input 对象的属性（比如 Value 属性）作为一个字符串变量的值。下表为同字符串对象有关的常用方法：

Table 2: 字符串对象的常用方法例举

常用方法	简要描述
strTemp.toLowerCase()	将字符串变成小写字母字符串
strTemp.toUpperCase()	将字符串变成大写字母字符串
strTemp.indexOf(abc)	字符串中是否存在 "abe"，有则返回一个正数，否则返回 -1
strTemp.substring(5,10)	截取字符串，从字符串的第 5 位开始，直到第 10 位
strTemp.replace("10", "a")	将 StrTemp 中的字符 "10" 用 "a" 替换
var strAry=StrTemp.split(",")	用 "," 作为分割符，将 StrTemp 的内容分割成数组 strAry
var n = parseInt("111")	将以数字开头的字符串变为数字

请看下列检测用户浏览器信息的例子：

```
function dispBrowser(){
      var userAgent = navigator.userAgent.toLowerCase();
      if (userAgent.indexOf("msie 6.0")==-1){
          alert("你的浏览器不是 IE6.0 的，本网页需要 IE6.0 才能得到理想显示效果！ ");
      }
}
```

简要注释：

如果用户的浏览器为 IE6.0 的版本，通过

```
var userAgent = navigator.userAgent.toLowerCase();
```

可以得到变量 userAgent 的值为“mozilla/4.0 (compatible; msie 6.0; windows nt 5.0; i-navfourf)”，通过 toLowercase()方法，将字符串转化为小写字母，有助于下面的判断字符串是否包含“msie 6.0”，不需要考虑大小写的问题。Navigator 对象是浏览器环境所构成的对象。

检查用户字符串是否是为整数：

```
function CheckNum(strTemp){
    if (!isNaN(strTemp)) {
            alert("包含有非数字字符！ ");
            return false;
        }else{
            var n = parseInt(strTemp);
                if (n.toString!= strTemp){
                alert("不是一个正数！ ");
                return false;
            }
        }
}
```

详细注释：

该函数用来检查字符串是否为“整数”，常用于检查用户输入的校验。比如，在一些分页显示的 Web 应用中，需要跳转到某页，此时用户的输入应该为整数数字字符，该函数可以判定用户输入的字符串是否为一个整数。函数中 isNaN()是 JavaScript 的内部函数，对参数进行运算，判断其是否为非数字。若参数为 NaN（注释：非数值值），则返回 true；否者返回 false；parseInt 也是一个 JavaScript 的内部函数，分析一个字符串，并返回给定基数或底数的一个整数（若没有给定基数，则假定基数为 10）。

对这种应用，还有一个办法就是，在用户输入的时候就限制用户只能输入数字字符（“事先预防机制”比“事后审核机制”是否更好一些呢？通过 JavaScript 来对用户录入进行有效性验证，对 Web 服务器来说，也是“事先预防机制”，防止不符要求，不完整的数据信息提交到服务器端），可以采用下面的 Input 对象：

```
<input type="text" name="PageNum"
    onkeypress="var k=event.keyCode; return k>=48&&k<=57||k==46"
    onpaste="return !clipboardData.getData('text').match(/\D/)"
    ondragenter="return false"
    style="ime-mode:Disabled"
>
```

上面语句中使用了正则表达式，关于正则表达式的运用，请参看后面“正则表达式的模式匹配”的内容。

3.4.2 日期对象

JavaScript 提供了特殊的内部对象－－Date 对象。利用该对象能够处理日期和时间信息。比如在网页上显示当前日期信息，或者依据不同的时段，给予用户不同的问候（上午好，还是晚上好？周末好？），也可以跟踪用户在网页上逗留了多长时间（如果你不怕麻烦，刻意去提供这种不太重要的功能的话）。

创建日期类型的对象的方法有：

1。 var d_today = new Date(); 创建一个包含当前时间和日期的 Data 对象；

2。通过 var d_time = new Date(2004,2,14,20,30,01) 表示创建一个 d_time 的 Data 对象，它的值为 2004 年 2 月 14 号 20 点 30 分 01 秒。其中的有些参数如果省略的话，JavaScript 会自动以零值补充。

像其他语言一样（比如 LotusScript 对于日期对象也有很多方法），JavaScript 对于日期对象也有相应的方法。以下是一些常用方法的举例：

Table 3: 日期对象的常用方法例举

常用方法	简要描述
getDate()	根据当地时间，返回指定日期的天

Table 3: 日期对象的常用方法例举

常用方法	简要描述
getDay()	根据当地时间，返回指定日期的星期几
getHours()	根据当地时间，返回指定日期的小时
getMonth()	根据当地时间，返回指定日期的月份
getMinutes()	根据当地时间，返回指定日期的分钟
getYear()	根据当地时间，返回指定日期的年份
setDate()	根据当地时间，设置日期变量的天
setHours()	根据当地时间，设置日期变量的小时
...	...

以下为一个格式化日期显示的例子，文件在随书光盘的 JavaScript 目录下的 JsTest4.htm，它的原来文件为：

```
<HTML><Head>
</Head><body><font color="red">
<SCRIPT language=javascript><!--
var isnMonths=new initArray("1 月","2 月","3 月","4 月","5 月","6 月","7 月","8 月","9 月","10 月","11 月","12 月")
var isnDays=new initArray("星期日","星期一","星期二","星期三","星期四","星期五","星期六","星期日");
today=new Date();

function initArray(){
        for(i=0;i<initArray.arguments.length;i++)
        this[i]=initArray.arguments[i];
}

function getFullYear(d){
        yr = d.getYear();
        if(yr<1000)         yr += 1900;
        return yr;
}
var StrTemp = getFullYear(today) +"年"  +" " + isnMonths[today.getMonth()];
StrTemp = StrTemp + "" + today.getDate()+"日 "+ isnDays[today.getDay()];
document.write(StrTemp);
//-->
</SCRIPT></font>
</body></HTML>
```

注释：

- JavaScript 函数中定义 isnMonths 和 isnDays 为两个数组对象。isnMonths[today.getMonth()]是通过 Date 对象中的月份值作为数组的下标，来访问数组中的值。
- 函数 initArray()用来构造一个数组，该函数使用了 JavaScript 的 Arguments 对象，Arguments 对象可以获得一个函数的所有参数集。
- 函数 initArray()中的 this 用来引用当前对象，this 是 JavaScript 中的一个关键字，this 是对当前对象的引用，在 JavaScript 由于对象的引用是多层次，多方位的，往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，最后自己都不知道现在引用的那一个对象，为此 JavaScript 提供了一个用于将对象指定当前对象的语句 this。

3.5 桢结构和框架窗口 (Frame,IFrame)

通过帧结构可以将几个 Web 页面集中在一个 URL 的地址下显示。大家一般通过帧结构（FrameSet）来组织站点的导航结构。比较典型的应用是：一个帧用于导航内容（目录或者大纲）的显示，它们的链接（具体的内容）在另外一个幅面较大的帧中显示。

IFrame 的作用也是用来组织页面显示，但是它可以任意嵌入到某个 Web 页面中间的任何位置。好像在一个 Web 页上“开天窗”一样，相比 FrameSet，FrameSet 有些像是“划豆腐块”一样。Iframe 的应用非常灵活。

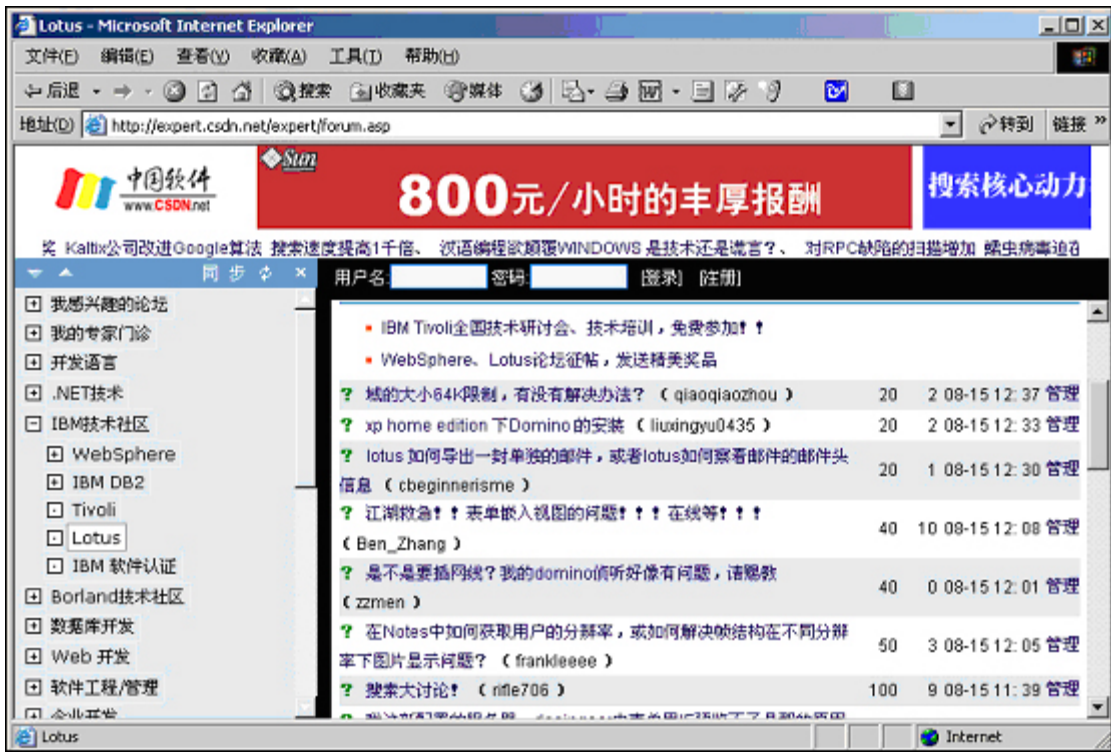
通过本小节的内容学习，我们主要来了解：

1。处于框架窗口中的 Web 页的 URL 的响应位置的确定－－了解 Target 属性

2。处于框架窗口中的 Web 页，如果通过 JavaScript 来访问其他框架（Frame、IFrame）中的对象；

下图是一个典型的包含三个帧结构（上，左，右）的浏览器页面：

Figure 5: 含有桢结构的网页文档



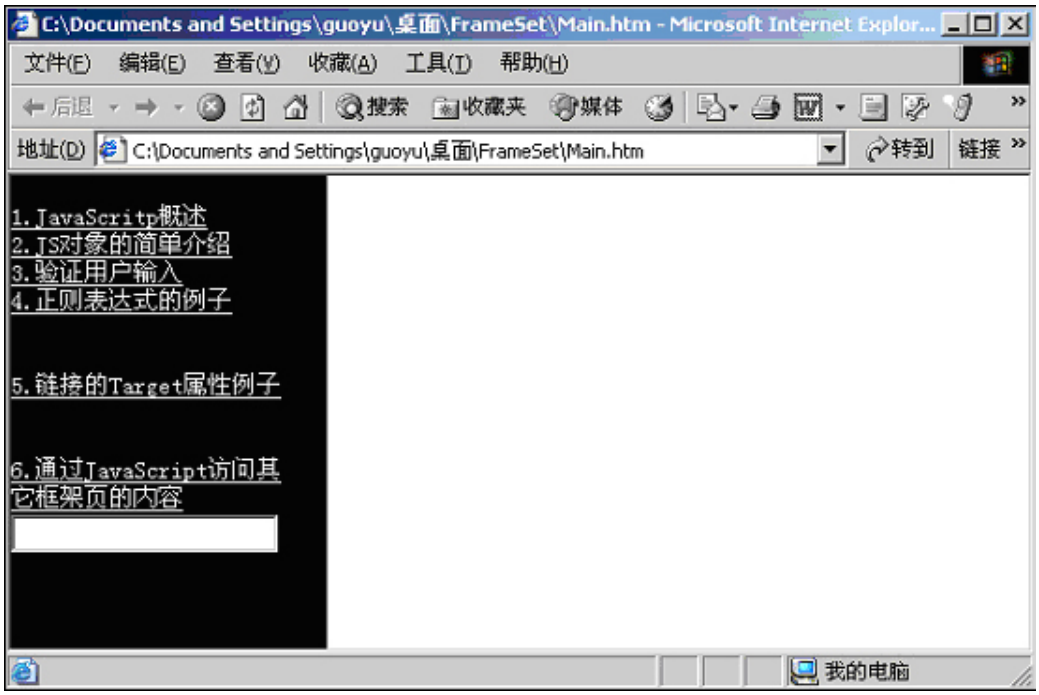
3.5.1 了解链接的 Target 属性

我们来具体分析一个例子：随书光盘的 JavaScript 目录下的 FrameSet 目录下的 Main.htm 文件源文件的如下：

```
<frameset name=framesetname cols="180,*"
          FRAMEBORDER=no
          border="0"
          framespacing="0">
      <frame name="Directory" src="left.htm"
           SCROLLING=AUTO
           NORESIZE=yes
           MarginHeight=0 MarginWidth=0
           leftMargin=0 target="right">
      <frame name="Content" src="right.htm"
           SCROLLING=AUTO
           NORESIZE=yes>
</frameset>
```

从上面代码可以看出，HTML 语句中定义了两个框架窗口，一个名称为 Directory，它对应于文件 left.htm，另一个框架窗口为 Content，它对应于文件 right.htm （该文件源代码不包含任何内容，为一张“空白”的 Web 页面）。Main.htm 的显示为：

Figure 6: 一个包含左右两个桢的例子



上图左边的帧 Directory 的 Src 指向的 Left.htm 页面的源文件如下：

```
<HTML><head>
<STYLE type=text/css>
        BODY {FONT: 14px 宋体;}
        a {color:#ffffff}
</style>
</head>
<body bgcolor="#000000"><br>
<a href="Js_1.htm" target="Content">1.JavaScritp 概述</a><br>
<a href="Js_2.htm" target="Content">2.JS 对象的简单介绍</a><br>
<a href="JsTest2.htm" target="Content">3.验证用户输入</a><br>
<a href="JsTest5.htm" target="Content">4.正则表达式的例子</a><br><br><br>
<a href="JsTest6.htm" target="Content">5.链接的 Target 属性例子</a><br><br><br>
<a href="JsTest7.htm" target="Content">6.通过 JavaScript 访问其它框架页的内容</a><br>
<input type="text" name="DispInfo">
</body>
</HTML>
```

注释：

在 HTML 中，链接的 HTML 标签有 Target 属性，该属性用来确定该链接的响应的位置。Target 属性的有效值可以是当前浏览器显示内容的任何框架窗口（Frame 或者 Iframe）的名称，本例子中，表示链接在名称为 Content 的帧中显示（即图 6 右边的空白处显示）。或者是下列值：

Table 4: 一些 Target 属性值

值	简要描述
_blank	在新的窗口中打开该链接
_parent	在本窗口的父窗口中打开该链接
_self	在本窗口中打开该链接
_top	在根窗口中打开该链接

请读者点击图 6 所示的 Main.htm 中的第 5 条“链接的 Target 属性例子”，该页面的链接，有四个链接，分别对应上面四种情况。

3.5.2 由 JavaScript，在不同的桢（窗口）间访问对象

如果一个浏览器显示内容包含多个（帧）Frame，在不同的（帧）Frame 中的 Web 页如何通过 JavaScript 访问其他帧中的 Web 页的对象呢？

请读者点击图 2-6 所示的 Main.htm 中的第 6 条“通过 JavaScript 访问其它框架页的内容”，该页面中有个按钮，该按钮 onclick 的所关联的 JavaScript 代码为：

```
top.Directory.document.forms[0].DispInfo.value ="测试测试";
```

实际上，上述内容也是**层层限定的逐步收缩法**。对于处于帧 Content 页面，top 代表浏览器的根窗口，该窗口中包含有 Directory 和 Content 窗口，top.Directory.document 表示：浏览器根窗口下的帧 Directory 所定义的窗口中的文档。在这里，强烈建议读者再次回到图 3 JavaScript 的文档对象模型图来对照分析上面代码。

对于不包含帧或者 Iframe 的窗口，我们往往直接通过 document 来作为对象访问的顶部层次，这意味着“当前窗口下的当前文档”。当浏览器窗口中包含有帧（或者 Iframe 时），而且需要从一个窗口访问另外一个帧所定义的窗口中的对象时，我们就需要在 document 之前，来限定究竟是访问哪个框架（Frame 或者 IFrame）下的文档。这类似于我们的邮件通信，当邮件的目的地也在同一个国家时，我们很少在前面加上国家的名称。如果你的邮件要发给其他国家的目的地，我们就需要指定国家的名称了。

3.5.3 了解 IFrame

Iframe 比帧结构集更加灵活，我们可以在一个 Web 页面的任何位置，加入 Iframe，在标签中设置参数（从设定大小，到指定边框样式等等），通过 SRC 的值来指定连接到某个 Web 页面。

请读者点击图 6 所示的 Main.htm 中的第 7 条“一个 Ifame 的例子”， 该链接对应的 Web 页 JsTest8.htm 中包含了一个 Iframe。它的源文件为：

```
<HTML><Head></Head>
<body>
    下面为一个 Ifame:<br>
    <IFRAME name=iFrame1 src="Js_2.htm"
            frameBorder=1 width=600 scrolling=auto height=400></IFRAME>
</body></HTML>
```

再请读者点击图 6 所示的 Main.htm 中的第 8 条“根据 Iframe 中页面内容的高度自适应调整 Ifame 高度的例子”， 该链接对应的 Web 页是 JsTest9.htm，在 JsTest9.htm 的 Iframe 的 Src 所指定的页面 Js_2_1.htm 比页面 Js_2.htm 的<head></head>之间增加了：

```
<SCRIPT language=javascript>
<!--
function toppage(){
  if (self.location!=top.location){
    parent.document.all(self.name).height = document.body.scrollHeight + 30;
    parent.document.all(self.name).width  = document.body.scrollWidth  + 30;
  }
}
function window.onload(){
        toppage();
        if (self.location!=top.location){
                window.resizeTo(document.body.scrollWidth,document.body.scrollHeight);
        }
}
// --></script>
```

简单注释：

- window.onload()当页面下载的完毕后会自动运行；
- window.onload()中的 if (self.location!=top.location)，表示当前窗口的页面是否等于根窗口的页面，如果不相等的话，表示该页处在一个框架中(Frame 或者 IFrame)，此时，将浏览器的窗口调整到当前文档的滑动条的高度。
- toppage()用来调整当前文档所在的窗口的宽度和高度，即是调整 Iframe 的宽度和高度；

4 正则表达式和模式匹配

正则表达式描述了字符串的一个模式，可以用来验证用户输入数据的格式。

正则表达式是一种可以用于模式匹配和替换的强有力的工具。我们可以在几乎所有的基于 UNIX 系统的工具中找到正则表达式的身影，例如，vi 编辑器，Perl 或 PHP 脚本语言，以及 awk 或 sed shell 程序等。此外，象 JavaScript 这种浏览器客户端的脚本语言也提供了对正则表达式的支持。由此可见，正则表达式已经超出了某种语言或某个系统的局限，成为人们广为接受的概念和功能。

正则表达式可以让用户通过使用一系列的特殊字符构建匹配模式，然后把匹配模式与数据文件、程序输入以及 WEB 页面的表单输入等目标对象进行比较，根据比较对象中是否包含匹配模式，执行相应的程序。

举例来说，正则表达式的一个最为普遍的应用就是用于验证用户在线输入的邮件地址的格式是否正确。如果通过正则表达式验证用户邮件地址的格式正确，用户所填写的表单信息将会被正常处理；反之，如果用户输入的邮件地址与正则表达的模式不匹配，将会弹出提示信息，要求用户重新输入正确的邮件地址。由此可见正则表达式在 WEB 应用的逻辑判断中具有举足轻重的作用。

4.1 定义正则表达式

类似于字符串被定义在引号内的字符，正则表达式被定义在一对“/”中间，所有字母表的字符和数字在正则表达式中自我匹配。比如，如果在表达式中包含有字母 a，JavaScript 将在由表达式验证的数据中查找 a。

```
var filter =/m$/
```

该句创建了一个正则表达式，可以匹配任意以字母 m 结尾的字符串。

其中位于“/”定界符之间的部分就是将要在目标对象中进行匹配的模式。用户只要把希望查找匹配对象的模式内容放入“/”定界符之间即可。为了能够使用户更加灵活的定制模式内容，正则表达式提供了专门的“元字符”。所谓元字符就是指那些在正则表达式中具有特殊意义的专用字符，可以用来规定其前导字符（即位于元字符前面的字符）在目标对象中的出现模式。

较为常用的元字符包括： “+”， “*”，以及 “?”。其中，“+”元字符规定其前导字符必须在目标对象中连续出现一次或多次，“*”元字符规定其前导字符必须在目标对象中出现零次或连续多次，而“?”元字符规定其前导对象必须在目标对象中连续出现零次或一次。

下面，就让我们来看一下正则表达式元字符的具体应用。

/fo+/

因为上述正则表达式中包含“+”元字符，表示可以与目标对象中的 “fool”， “fo”， 或者 “football”等在字母 f 后面连续出现一个或多个字母 o 的字符串相匹配。

/eg*/

因为上述正则表达式中包含“*”元字符，表示可以与目标对象中的 “easy”， “ego”， 或者 “egg”等在字母 e 后面连续出现零个或多个字母 g 的字符串相匹配。

/Wil?/

因为上述正则表达式中包含“？”元字符，表示可以与目标对象中的 “Win”， 或者 “Wilson”,等在字母 i 后面连续出现零个或一个字母 l 的字符串相匹配。

除了元字符之外，用户还可以精确指定模式在匹配对象中出现的频率。例如，

/jim{2,6}/

上述正则表达式规定字符 m 可以在匹配对象中连续出现 2-6 次，因此，上述正则表达式可以同 jimmy 或 jimmmmmy 等字符串相匹配。

在对如何使用正则表达式有了初步了解之后，我们来看一下其它几个重要的元字符的使用方式。

\s：用于匹配单个空格符，包括 tab 键和换行符；
\S：用于匹配除单个空格符之外的所有字符；
\d：用于匹配从 0 到 9 的数字；
\w：用于匹配字母，数字或下划线字符；
\W：用于匹配所有与\w 不匹配的字符；
（说明：我们可以把\s 和\S 以及\w 和\W 看作互为逆运算）

除了我们以上所介绍的元字符之外，正则表达式中还具有另外一种较为独特的专用字符，即定位符。定位符用于规定匹配模式在目标对象中的出现位置。

较为常用的定位符包括： “^”， “$”， “\b” 以及 “\B”。其中，“^”定位符规定匹配模式必须出现在目标字符串的开头，“$”定位符规定匹配模式必须出现在目标对象的结尾，\b 定位符规定匹配模式必须出现在目标字符串的开头或结尾的两个边界之一，而“\B”定位符则规定匹配对象必须位于目标字符串的开头和结尾两个边界之内，即匹配对象既不能作为目标字符串的开头，也不能作为目标字符串的结尾。同样，我们也可以把“^”和“$”以及“\b”和“\B”看作是互为逆运算的两组定位符。举例来说：

```
/^hell/
```

因为上述正则表达式中包含“^”定位符，所以可以与目标对象中以 “hell”， “hello”或“hellhound”开头的字符串相匹配。

```
/ar$/
```

因为上述正则表达式中包含“$”定位符，所以可以与目标对象中以 “car”， “bar”或 “ar” 结尾的字符串相匹配。

```
/\bbom/
```

因为上述正则表达式模式以“\b”定位符开头，所以可以与目标对象中以 “bomb”， 或 “bom”开头的字符串相匹配。

```
/man\b/
```

因为上述正则表达式模式以“\b”定位符结尾，所以可以与目标对象中以 “human”， “woman”或“man”结尾的字符串相匹配。

4.2 字符类

字符类是括在方括号中的文字字符组合。因此，正则表达式/[xyz]/可以匹配任意包括 x、y、z 中一个字符。在字符类中经常出现下列符号“^”、“-”、“|”符号。如下例子：

```
/[^A-C]/
```

上述字符串将会与目标对象中除 A，B，和 C 之外的任何字符相匹配。一般来说，当“^”出现在 “[]”内时就被视做否定运算符；而当“^”位于“[]”之外，或没有“[]”时，则应当被视做定位符。

```
/Th\*/
```

上述正则表达式将会与目标对象中的“Th*”而非“The”等相匹配。反斜杠“\”表示转义字符序列，比如“*”表示字符*，“\n”表示换行。

下列式子表示可以匹配 3 个数字或者 4 个小写字母：

```
var filter =/\d{3}|[a-z]{4}/;
```

4.3 正则表达式的应用例子

我们可以使用正则表达式的 test 或者 search 方法来发现字符中是否符合某个模式。比如：

```
var ResultFlag = forms[0].UserName.value.search(/[a-z]|[A-Z]/);
if (ResultFlag==-1){alert(表单中的 UserName 没有包含任何字符！)};
```

下列为一个检查用户邮件地址格式的例子，该例子是随书光盘的 JavaScirpt 目录下的 JsTest5.htm。

```
<HTML><head>
<script language="Javascript"> <!--
function vertifyMailAddress(obj){
var email = obj.email.value;
        var pattern = /^([a-zA-Z0-9_-])+@([a-zA-Z0-9_-])+(\.[a-zA-Z0-9_-])+/;
         flag = pattern.test(email);
         if(flag){
                alert("恭喜！邮件地址输入正确，表单将提交！"); return true;
         }else{
                alert("失败！邮件地址格式不正确!表单不会提交"); return false;
         }
}
--></script>
</head><body>
<form onSubmit="return vertifyMailAddress(this);">
<input name="email" type="text">
<input type="submit">
</form>
</body></HTML>
```

5　可参考学习的，精美的代码例子

在本文档的“2.1 JavaScript 操作对象的简单介绍—属性和方法”小节中，我们提到 JavaScript 访问的对象由三个来源：

1。浏览器环境和 Html 语句所构成的现成对象（JavaScript 的文档对象模型图 DOM）；
2。JavaScript 内置类所创建的对象；
3。用户自己通过 JavaScript 编写类的代码，并创建类的实例---对象。

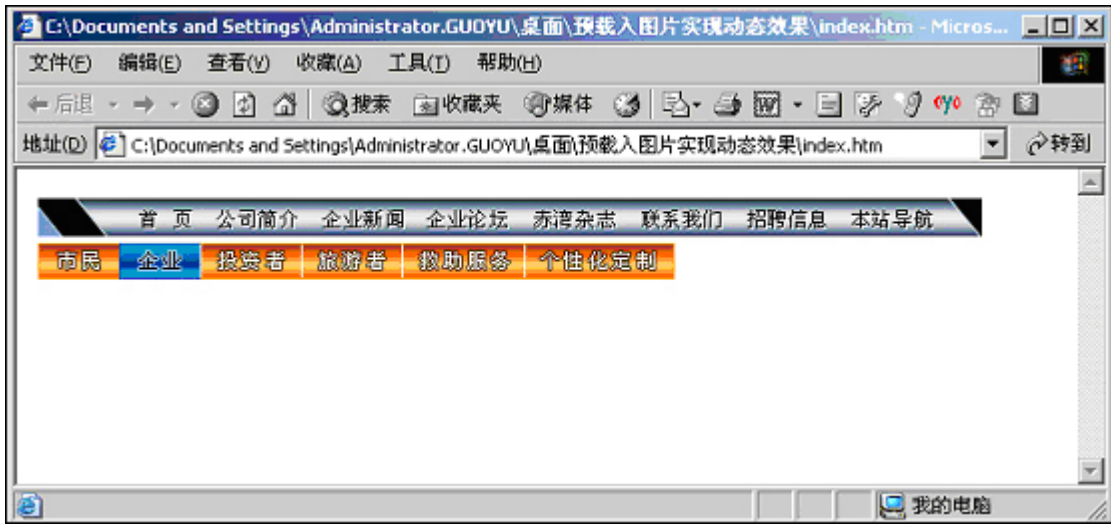
在前面部分中，我们主要讨论了前两种访问对象的来源。在这里，我们将会遇到一些第三种情况的例子。在本部分的内容中，我们将深入探索一些复杂的，优美的 JavaScript 代码。你可以在这些代码的基础上进行个性化的修改，来适合你的 WEB 应用程序设计的需要。下面部分提到的树图、下拉菜单、日历的代码来自国外的共享软件作者的成果，感谢原作者的杰出劳动。

在前面也提到过：**JavaScript 是“浏览器”上的程序语言，当 Web 服务器输出内容（包括 JavaScript 的程序代码）到浏览器时，此时，JavaScript 可以操纵浏览器上的一切内容**。通过本部分的例子，你将有一些很好的体会。

5.1　预载入图片，实现导航按钮的动态效果

很多 Web 站点在首页的上部区域会建立导航栏，为了使导航栏更加美观，有些会采用“图片按钮”，并且会采用动态效果：当鼠标移到导航按钮上时，该导航按钮的图片发生变化，到鼠标移开该导航按钮时，恢复到最初的图片。下图为两个导航条，图片所示的情形为鼠标移动到“企业”导航按钮的情景。

Figure 7: 动态变换图片的按钮的网页



在 CD 上：

查阅随书光盘上的 JavaScript 目录下的“预载入图片实现动态效果”目录中的文件是上图显示所用的文件。

这是如何实现的呢？首先我们要保证每个导航按钮有对应的存在一些差异的两个图片可用的前提条件。另外我们知道在 Html 中表示图片的标签语句类似于：

<IMG name=W_Image1 src="images/w_wen_1.gif" height=22 width=49 border=0 >

上述语句表示一个图片对象，它的名称为 W_Image1，它的 Src 属性为 images/w_wen_1.gif。我们可以通过 JavaScript 来访问该图片对象 W_Image1，当鼠标移动到该导航条（关联事件 onMouseOver）时，通过 onMouseOver 所关联的 JavaScript 代码，更改该图片对象的 Src 属性，也就是说图片对象 W_Image1 指向另外一个图片（导航条的外观就发生变化）。如果鼠标移开到该导航条（关联事件 onMouseOut）时，通过 onMouseOver 所关联的 JavaScript 代码，更改该图片对象的 Src 属性，恢复为最初的图片位置。

下面为图 7 的 index.htm 的主要代码（为了节省篇幅，只选取一个导航按钮）

```
<HTML><Head>
<script language="JavaScript" src="pic_change.js"></script>
</Script></Head>
<body>
<table border=0><tr>
<TD vAlign=center width="50%" height=25>
        <A class=bt  onmouseover="MM_swapImage('W_Image1','images/w_wen_11.gif')"
        onmouseout=MM_swapImgRestore()>
        <IMG name=W_Image1 src="images/w_wen_1.gif" width=49  height=22border=0>
        </A>
</TD></tr>
</table>
</body>
```

注释：

上述 Html 语句中，MM_swapImage('W_Image1','images/w_wen_11.gif')是用 images/w_wen_11.gif 变化图像对象 W_Image1 的 Src 属性的函数。而 MM_swapImgRestore()是恢复原来 Src 属性的函数，具体的函数在 pic_change.js 库文件中。

库文件 pic_change.js 的代码为：

```
function MM_preloadImages(){
        if(document.images){
                var imgFiles = MM_preloadImages.arguments;
                var preloadArray = new Array();
                for (var i=0;i< imgFiles.length;i++){
                        preloadArray[i] = new Image;
                        preloadArray[i].src = imgFiles[i];
                }
        }
}

function MM_swapImgRestore() {
        var i,x,a=document.MM_sr;
        for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++)
                x.src=x.oSrc;
}
function MM_findObj(n, d) {
        var p,i,x;
        if(!d) d=document;
        if((p=n.indexOf("?"))>0&&parent.frames.length) {
                d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
        if(!(x=d[n])&&d.all)
```

```
                x=d.all[n];
        for (i=0;!x&&i<d.forms.length;i++)
                x=d.forms[i][n];
        for(i=0;!x&&d.layers&&i<d.layers.length;i++)
                x=MM_findObj(n,d.layers[i].document);
        if(!x && d.getElementById)
                x=d.getElementById(n);
        return x;
}

function MM_swapImage(){
        var i,j=0,x,a=MM_swapImage.arguments;
        document.MM_sr=new Array;    //定义一 document 下的 MM_sr 的数组对象
    for(i=0;i<a.length;i++){
                if ((x=MM_findObj(a[i]))!=null){
                        document.MM_sr[j++]=x;
                        if(!x.oSrc) x.oSrc=x.src; x.src=a[i+1];
                }
        }
}
```

详细注释：

1. 我们先来看看变换图片的 Src 属性的 MM_swapImage()函数。通过 a=MM_swapImage.arguments 语句，调用函数的 arguments 对象（这是 JavaScript 的内置对象，使用于函数当中，返回函数的参数数组），将函数的参数作为数组 a 的值。x=MM_findObj(a[i])，该语句是将函数 MM_swapImage()的第一个参数作为函数 MM_findObj()的参数，函数 MM_findObj()的作用是：根据表示 name 或者 Id 属性值的参数值来找到对应的对象（函数 MM_findObj()的作用不错哦！很有用武之地）。

   语句：

   ```
     document.MM_sr[j++]=x;
   if(!x.oSrc) x.oSrc=x.src; x.src=a[i+1];
   ```

   的含义：前一句代表一个图片对象的 x 存放到数组 MM_sr 中，后一句表示，将图片对象 x 的 src 属性的值存放到 oSrc 属性中。而将 MM_swapImage()的第 2 个参数的值作为图片对象 x 的 Src 属性值，从而使鼠标移到导航按钮上时，图片发生变化。
2. 函数 MM_swapImgRestore()的作用是恢复图片原有的 Src 属性，从语句 x.src=x.oSrc;可以清楚的看出来，在 MM_swapImage()函数中，将最初的图像的 Src 属性的值存放在 oSrc 中，这里重新取回 oSrc 的值赋给 src 属性。
3. 函数 MM_preloadImages()并不是必需的，但建议使用该函数，该函数的作用是，将一些表示图片位置的参数，通过函数 MM_preloadImages()，生成一个表示图片对象的数组。通过这个过程，可以预先下载那些隐含的图片，当用户鼠标放在导航的按钮上时，从而不必从 Web 服务器上逐次下载图片。提高响应速度。该函数的使用方法在 Html 的<body></body>标签之间加入，其中 strTemp 表示那些隐藏的图片的位置值。

   ```
   <script language="JavaScript"> <!—
                   var strTemp = "'images/button_01t.gif','images/button_02t.gif','images/button_03t.gif',";
                   strTemp = strTemp + "'images/button_04t.gif','images/button_05t.gif','images/button_06t.gif',"
                   MM_preloadImages(strTemp);
   //--></SCRIPT>
   ```

4. 本语句：document.MM_sr=new Array; 是定义一个 document 对象下的变量 MM_sr，该变量是一个数组。另外在这里顺便提一下：在 JavaScript 的函数当中，如果这样定义变量：

   var strTemp=”This is a string”;

   表示该变量是函数内部的变量－－局部变量；如果这样定义：

   strTemp=”This is a string”;

   表示是全局变量，在其他的函数中可以直接访问；

5.2 显示对象的提示信息

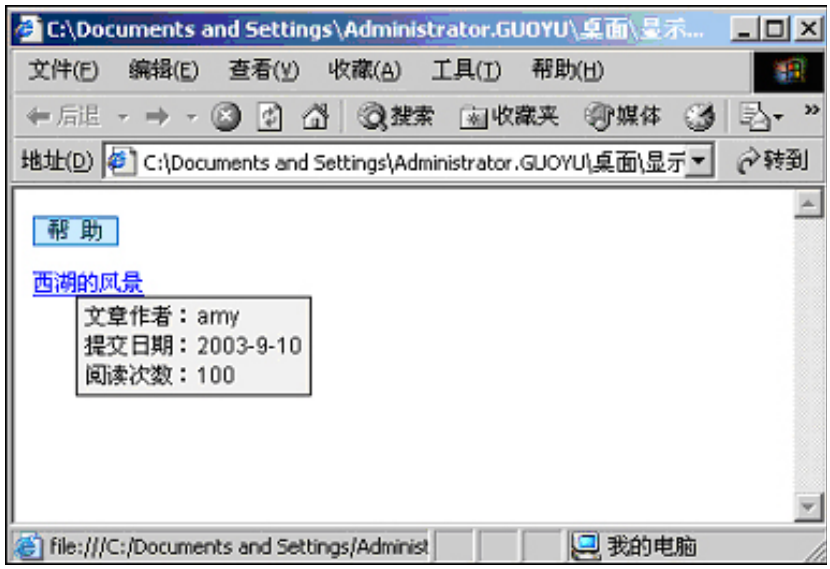
在一般的 Windows 应用程序中，如果鼠标停留在某个按钮上一会儿。将会弹出一些黄色背景的提示信息框，比如：我们在使用 MS Word 作文档处理时，如果将鼠标放在 Word 的工具条上的表格按钮上，将会弹出“插入表格”的提示信息。

在 Html 中的对象中，也会存在这个特性，只要我们在写对象的 Html 标签语句的时候，中间增加一个 Title 的属性。比如：

```
<INPUT title=查看帮助信息 type=button value=帮助 name=btnHtmHelp>
```

如果我们需要进一步改进这种显示提醒信息的方法，比如背景颜色，显示时间，半透明的渐隐效果等等。我们可以通过 JavaScript 来做到，如下图：

Figure 8: 显示对象的提示信息的网页



在 CD 上：

查阅随书光盘上的 JavaScript 目录下的“显示对象的提示信息”目录中的文件是图 8 显示所用的文件。

上图的 Show.htm 文件的源文件为：

```
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=gb2312">
<STYLE type=text/css>
BODY{FONT: 11px Verdana, Arial, Helvetica, sans-serif,宋体; MARGIN-LEFT: 10px;}
.btnmenuview {
        BORDER-RIGHT: #0066cc 1px solid; PADDING-RIGHT: 1px; BORDER-TOP: #0066cc 1px solid;
        PADDING-LEFT: 1px; BACKGROUND: #c4e5fa; PADDING-BOTTOM: 0px;
        FONT: 12px "宋体"; BORDER-LEFT: #0066cc 1px solid; CURSOR: hand; COLOR: #003048;
        PADDING-TOP: 0px; BORDER-BOTTOM: #0066cc 1px solid; LETTER-SPACING: 6px; HEIGHT: 16px
        }
.btnmenuviewmouseover{
        BORDER-RIGHT: #0066cc 1px outset; PADDING-RIGHT: 1px; BORDER-TOP: #0066cc 1px outset;
        PADDING-LEFT: 1px; BACKGROUND: #62b6f0; PADDING-BOTTOM: 0px; FONT: 12px "宋体";
        BORDER-LEFT: #0066cc 1px outset; CURSOR: hand; COLOR: #003048; PADDING-TOP: 0px;
        BORDER-BOTTOM: #0066cc 1px outset; LETTER-SPACING: 6px; HEIGHT: 16px
        }
}
</STYLE>
<script language="JavaScript" src="showtext.js"></script>
</HEAD>
<BODY>
<INPUT class=btnmenuview onmouseover="this.className='btnmenuviewmouseover'"
    onmouseout="this.className='btnmenuview'"  title=查看帮助信息  type=button value=帮助
    name=btnHtmHelp>
<br><br>
<a href="#a" alt="文章作者：amy<br>提交日期：2003-9-10<br>阅读次数：100<br>">西湖的风景</a>
</body>
</html>
```

上述文件引入外部的 showtext.js 库文件，该库实现以特定效果显示对象的 Ttile 或者 alt 属性的值，showtext.js 文件程序清单：

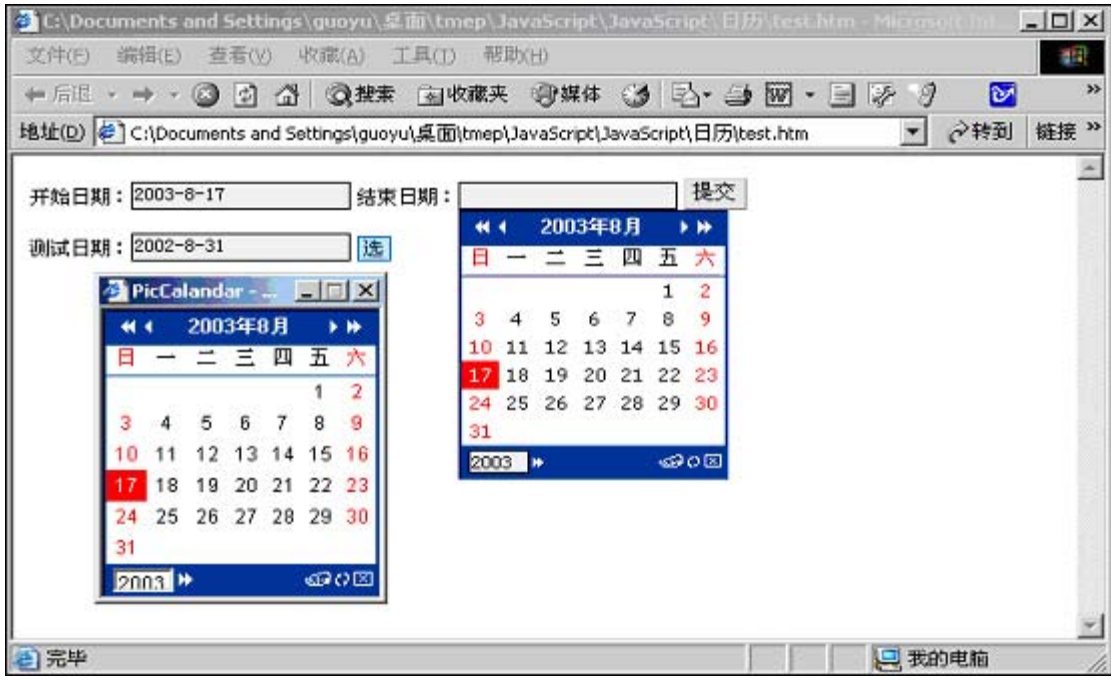
```
//***********默认设置定义.*********************
tPopWait=50;                //停留 Wait 豪秒后显示提示。
tPopShow=4000; //显示 tShow 豪秒后关闭提示
showPopStep=20;
popOpacity=85;   //设置透明度百分比，100 表示不透明

//***************内部变量定义*****************
sPop=null;
curShow=null;
tFadeOut=null;
tFadeIn=null;
tFadeWaiting=null;

//***************样式表内容*****************
document.write("<style type='text/css'id='defaultPopStyle'>");
document.write(".cPopText {  background-color: #F8F8F5;color:#000000; border: 1px #000000 solid; font-
size:12px;font-family:Arial;padding-right: 4px; padding-left: 4px; height: 20px; padding-top: 2px; padding-
bottom: 2px; filter: Alpha(Opacity=0)}");
document.write("</style>");
document.write("<div id='dypopLayer' style='position:absolute;z-index:1000;' class='cPopText'></div>");
```

```
//***************显示提示信息的主程序*****************
function showPopupText(){
var o=event.srcElement;
        MouseX=event.x;
        MouseY=event.y;
        if(o.alt!=null && o.alt!=""){o.dypop=o.alt;o.alt=""};

          //如果下行不注销，则对元素的 Title 属性的显示特性也作变化。
          //if(o.title!=null && o.title!=""){o.dypop=o.title;o.title=""};
          if(o.dypop!=sPop) {
                        sPop=o.dypop;
                        clearTimeout(curShow);
                        clearTimeout(tFadeOut);
                        clearTimeout(tFadeIn);
                        clearTimeout(tFadeWaiting);
                        if(sPop==null || sPop=="") {
                                dypopLayer.innerHTML="";
                                dypopLayer.style.filter="Alpha()";
                                dypopLayer.filters.Alpha.opacity=0;
                                }
                        else {
                                if(o.dyclass!=null) popStyle=o.dyclass
                                        else popStyle="cPopText";
                                curShow=setTimeout("showIt()",tPopWait);
                        }
        }
}

function showIt(){
                dypopLayer.className=popStyle;
                dypopLayer.innerHTML=sPop;
                popWidth=dypopLayer.clientWidth;
                popHeight=dypopLayer.clientHeight;
                if(MouseX+12+popWidth>document.body.clientWidth) popLeftAdjust=-popWidth-24
                        else popLeftAdjust=0;
                if(MouseY+12+popHeight>document.body.clientHeight) popTopAdjust=-popHeight-24
                        else popTopAdjust=0;
                dypopLayer.style.left=MouseX+12+document.body.scrollLeft+popLeftAdjust;
                dypopLayer.style.top=MouseY+12+document.body.scrollTop+popTopAdjust;
                dypopLayer.style.filter="Alpha(Opacity=0)";
                fadeOut();
}

function fadeOut(){
        if(dypopLayer.filters.Alpha.opacity<popOpacity) {
                dypopLayer.filters.Alpha.opacity+=showPopStep;
                tFadeOut=setTimeout("fadeOut()",1);
                }else {
                        dypopLayer.filters.Alpha.opacity=popOpacity;
                        tFadeWaiting=setTimeout("fadeIn()",tPopShow);
                        }
}

function fadeIn(){
        if(dypopLayer.filters.Alpha.opacity>0) {
                dypopLayer.filters.Alpha.opacity-=1;
                tFadeIn=setTimeout("fadeIn()",1);
                }
}
document.onmouseover=showPopupText;
```

5.3 一个精美的日历

在 Domino 的域的类型中有日期时间类型，但在基于 Web 的设计中，并没有日期或者时间的表单对象。但在 Web 应用程序当中，有时候需要提供这些内容，既方便用户输入日期和时间信息，同时也限定了用户输入的内容是符合日期和时间的格式的。如下图：

Figure 9: 精美的日历的例子



在 CD 上：

查阅随书光盘上的 JavaScript 目录下的“日历”目录中的文件是图 9 显示所用的文件。

上图主要的 Html 源文件为（省略了一些内容）：

```
<script src="Calendar.js" language="javascript"></script>
</head><BODY onclick="hiddenCal()">
<form name="Searchform" method="post"   action="#a" onsubmit="return checkDate(this)">
开始日期：<INPUT name=StartDate TYPE="TEXT" onfocus="showCalendar(this);this.blur();"
class="boxinput" onmouseout="calshow=false"/>
结束日期：<INPUT name=EndDate  TYPE="TEXT" onfocus="showCalendar(this);this.blur();"
class="boxinput" onmouseout="calshow=false"/>
<input type="submit" name="Submit" value="提交" class="button">
<div id="calendar" style="position:absolute;height:140px;width:160px;display:none;border:1px inset #003399;"
onmouseover="calshow=true" onmouseout="calshow=false"/></div>
```

简要注释：
通过
<script src="Calendar.js" language="javascript"></script>
引入日历的 JavaScript 的具体的代码。因为该代码较长，有 300 行左右。请阅读随书光盘上的 Calendar.js 文件。

当我们理解清楚 Calendar.js 文件后，可以根据我们的需要对文件进行适当的修改，比如图 9 中的测试日期的时间输入，通过点击一个“选”的按钮，打开“picCalendar”的窗口，在该窗口的进行日期的选择，选择的值将赋给原窗口的表示“测试日期”的文本输入框，这主要是修改 Calendar.js 的中代表输入文本框的对象 fillInput，使它：

fillInput = window.opener.document.forms[0].TestDate;

详细的源文件请参考光盘目录：JavaScript\日历\popWindow\CalCalendar2.js，并请将它同 Calendar.js 做比较。

5.4 一个下拉菜单

菜单是 C/S(Client/Server)方式下的应用程序设计，必不可缺的设计内容。通过菜单和菜单的分支，将一些相关的功能进行分类收集在一个菜单项下。对众多的功能进行层次性的分类组织，从而给用户带来方便性。良好的、符合用户习惯的菜单设计也是出色的用户界面设计的一个内容。

对基于 Domino 群件系统的办公自动化应用来说，依据不同企业员工数量和业务规模，所应用的数据库模块数量并不一样。作者对所在企业（400 多人研发人员）的原有的办公自动化系统进行 Web 时，同时对原有的数据库模块进行系统整顿，包括流程变更、废弃或者新增一些应用，最后保留下来的数据库流程应用大概有 40 多个，文档资料库 30 多个。而作者了解到国内另外一家具有 15000 人的通信研发企业，多年下来共作了 2000 多个数据库应用。如果将这些数据库进行分类整合，提供一个方便的访问入口，在 Domino 的 Web 设计中，显得非常重要。

借鉴 C/S 下的菜单的设计，在 Web 设计当中，也提供类似的下拉菜单设计，是一个可供选择的方案。我们来讨论这种方法的一个实例。如图：

Figure 10: 一个下拉菜单的网页例子

在 CD 上：

查阅随书光盘上的 JavaScript 目录下的“下拉菜单”目录中的文件是图 10 显示所用的文件。

图 10 中的 index.htm 的源文件为（省略一部分内容）：

```
<html>
<head>
<title>信息系统</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link href="default.css" rel="stylesheet" type="text/css">
<script language="JavaScript1.2" src="menu_script.js"></script>
<script language="JavaScript1.2" src="menu_data.js"></script>
</head>

<body background="images/Bodybg1.gif" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<table width="770" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="599" height="22" bgcolor="#000000"> <div align="center">
      <table width="87%" height="22" border="0" cellpadding="0" cellspacing="0">
        <tr class="whitenav">

                                <td width="16%" onmouseout="this.style.backgroundColor="
    "onmouseover=this.style.backgroundColor='#C60400'>
    <div align="center"><span class="font11">
    <a href="#a" class="whitenav">首页</a></span></div></td>

          <td width="14%" onmouseout="this.style.backgroundColor=""
          onmouseover=this.style.backgroundColor='#C60400' >
    <div align="center"><span class="font11">
        <a href="/webdesign/" class="whitenav" id="menu0"
            onmouseover="showMenu(event, 0)" onmouseout="hideMenu(event, 0)">
办公管理</a></span></div></td>

          <td width="14%" onmouseout="this.style.backgroundColor=""
          onmouseover=this.style.backgroundColor='#C60400'>
<div align="center"><span class="font11">
    <a href="/wpm/" class="whitenav" id="menu1"
       onmouseover="showMenu(event, 1)" onmouseout="hideMenu(event, 1)">
项目管理</a></span></div></td>
                    //..... 这里省略了一些雷同上面的代码
        </tr>
      </table>
    </div></td>
  <td width="171" bgcolor="#C60400"><div align="center"><font color="#FFFFFF">
      </font></div></td>
  </tr>
</table>
</body>
</html>
```

注释：

上面的源代码结构简单，比较清楚。我们主要来理解：

<script language="JavaScript1.2" src="menu_script.js"></script>

<script language="JavaScript1.2" src="menu_data.js"></script>

通过上述代码引入外部的 JavaScript 库文件。我们提起过，把一些经过严格测试过的 Javascript 函数作为 js 库文件的好处：提高 Js 代码的稳定性，比放在 Html 中不容易出错，同时减少了 Html 文件的长度；作为库文件，被 Web 页引用过一次，其它 Web 页也引用时，直接从用户计算机的缓存中取得该库，减少了网络流量，速度得到提高。

menu_script.js 是实现下拉的菜单的 JavaScript“程序”文件（它比较稳定，它的作者经过了严格测试，我们可以放心使用，不必要去考究其中细节）。我们主要要了解的是 **menu_data.js**，它是菜单的数据配置文件。

以下内容为 menu_data.js 文件的主要内容的节选，并作相应的注释，对：

```
/*-------------------------------------------
对菜单的一些设置
-------------------------------------------*/
  DQM_sub_menu_width = 130;
  DQM_urltarget = "_self"       //点击菜单项目，其链接的 Target 属性设定
  DQM_onload_statement = ""
  DQM_codebase = ""
  DQM_border_color = "#000000";    // 子菜单表格的颜色
  DQM_menu_bgcolor = "#eeeeee";   //鼠标停留在子菜单表格的单元格时，单元格的颜色；
  DQM_hl_bgcolor = "#dddddd";      //子菜单表格的单元格的颜色；

  DQM_sub_xy = "0,0";
  DQM_border_width = 1;   //子菜单表格的边框的宽度；
  DQM_divider_height = 1;  //子菜单单元格之间的间隔宽度；

/*-------------------------------------------
菜单中字体的设置
-------------------------------------------*/
  DQM_textcolor = "#333333"  //字体颜色
  DQM_fontfamily = "Verdana"  //字体
  DQM_fontsize = 11          //字体大小
  DQM_fontsize_ie4 = 9
  DQM_textdecoration = "normal"
  DQM_fontweight = "normal"
  DQM_fontstyle = "normal"
  DQM_hl_textcolor = "#000000"
  DQM_hl_textdecoration = "underline"
  DQM_margin_top = 3
  DQM_margin_bottom = 3
  DQM_margin_left = 5


/*-------------------------------------------
菜单上的图片预定义 （这里只定义两组图片，分别为 0 和 1）
-------------------------------------------*/
  DQM_icon_image0 = "images/bullet.gif"      //图片的位置
  DQM_icon_rollover0 = "images/bullet_hl.gif"  //当鼠标移动到该子菜单项目时，替换图片的位置。
  DQM_icon_image_wh0 = "13,8"           //图片的大小

  DQM_icon_image1 = "images/arrow.gif"
  DQM_icon_rollover1 = "images/arrow.gif"
  DQM_icon_image_wh1 = "13,10"
```

```
/*-------------------------------------------
子菜单分支的设置
-------------------------------------------*/
DQM_sub_xy0 = "-11.5,15"        //第一组菜单（以 0 为开始）的位置定位
DQM_sub_menu_width0 = 150;     //第一组菜单（以 0 为开始）的宽度定义
DQM_subdesc0_0 = "公告通知"    //第一组菜单的第一个子菜单分支（以 0-0 标志）的显示文字
DQM_subdesc0_1 = "热点讨论"    //第一组菜单的第二个子菜单分支（以 0-1 标志）的显示文字
DQM_subdesc0_2 = "请假申请"
DQM_subdesc0_3 = "加班申请"
DQM_subdesc0_4 = "用车申请"
DQM_subdesc0_5 = "公文发布"
DQM_subdesc0_6 = "呵呵呵呵"

DQM_icon_index0_0 = 0    //第一组菜单的第一个子菜单分支（以 0-0 标志）所使用的图片组
DQM_icon_index0_1 = 0    //这里都使用“菜单上的图片预定义”的第一组图片
DQM_icon_index0_2 = 0
DQM_icon_index0_3 = 0
DQM_icon_index0_4 = 0
DQM_icon_index0_5 = 0
DQM_icon_index0_6 = 0

DQM_url0_0 = "/webdesign/2/index_1.htm"  //第一组菜单第一个子菜单分支的链接的文件；
DQM_url0_1 = "/webdesign/3/index_1.htm"  //第一组菜单第二个子菜单分支的链接的文件；
DQM_url0_2 = "/webdesign/4/index_1.htm"
DQM_url0_3 = "/webdesign/5/index_1.htm"
DQM_url0_4 = "/webdesign/6/index_1.htm"
DQM_url0_5 = "/webdesign/7/index_1.htm"
DQM_url0_6 = "/webdesign/8/index_1.htm"

//....以下省略了其他四组菜单分支的设置
```


在 index.htm 页面中的：

```
<a href="/webdesign/" class="whitenav" id="menu0"
            onmouseover="showMenu(event, 0)" onmouseout="hideMenu(event, 0)">
办公管理</a>
```

通过 id="menu0"来表示“办公管理”和 menu_data.js 文件中的第一组菜单对应。函数 showMenu(event, 0)和 hideMenu(event, 0)中的参数 0，对应第一组菜单的显示和隐藏。

另外要注意的一点是：menu_ie.js 文件在 menu_script.js 文件中使用到，menu_script.js 文件中有一句：

document.write("<script language=\"JavaScript1.2\" src=\"menu_"+brn+".js\"></script>")

而 menu_script.js 被 index.htm 引用， 写法 src=\"menu_"+brn+".js\"表示 menu_ie.js 和 Index.htm 必须要在同一个目录下，Index.htm 才能够找到 menu_ie.js 文件。

假如 menu_ie.js 文件放在 Index.htm 文件所在的目录下的 JsScript 目录下，那么 menu_script.js 文件的引用 menu_ie.js 的语句应该变化为：

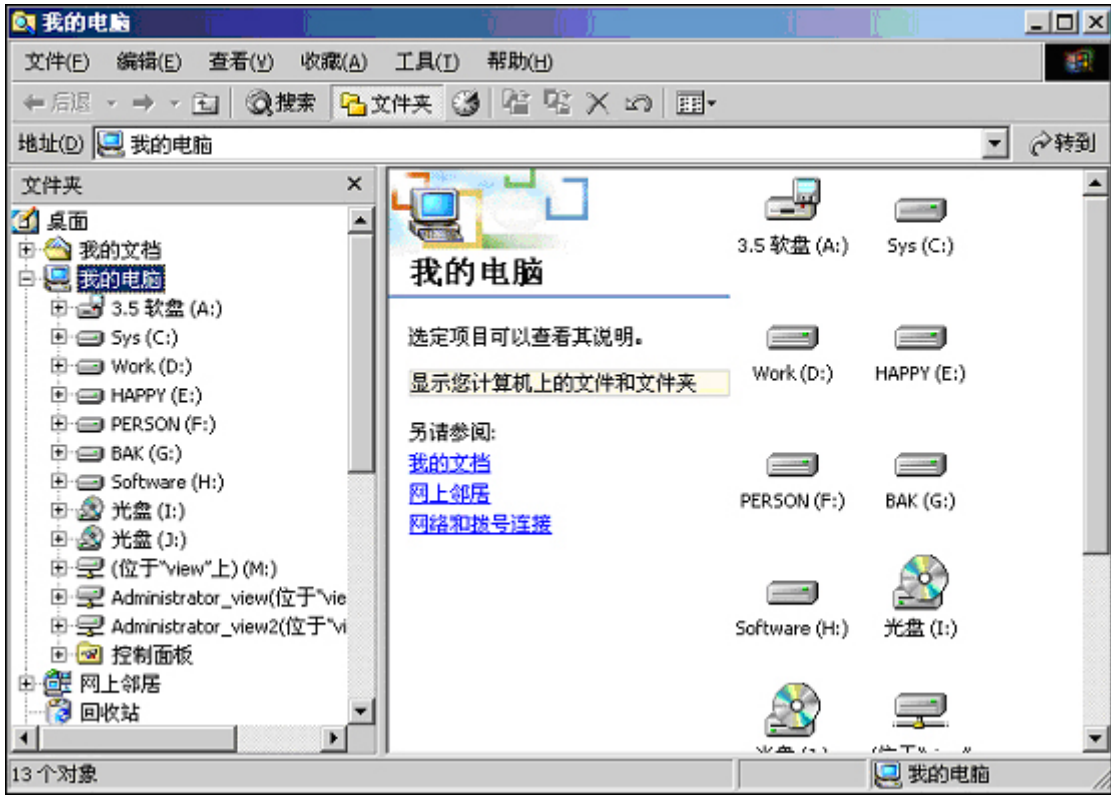
document.write("<script language=\"JavaScript1.2\"
src=\"JsScript/menu_"+brn+".js\"></script>")

总之，当我们在 Html 中间引用外部 js 库文件时，需要确保外部 js 库文件的正确的路径写法。

5.5 类似于资源管理器的树图

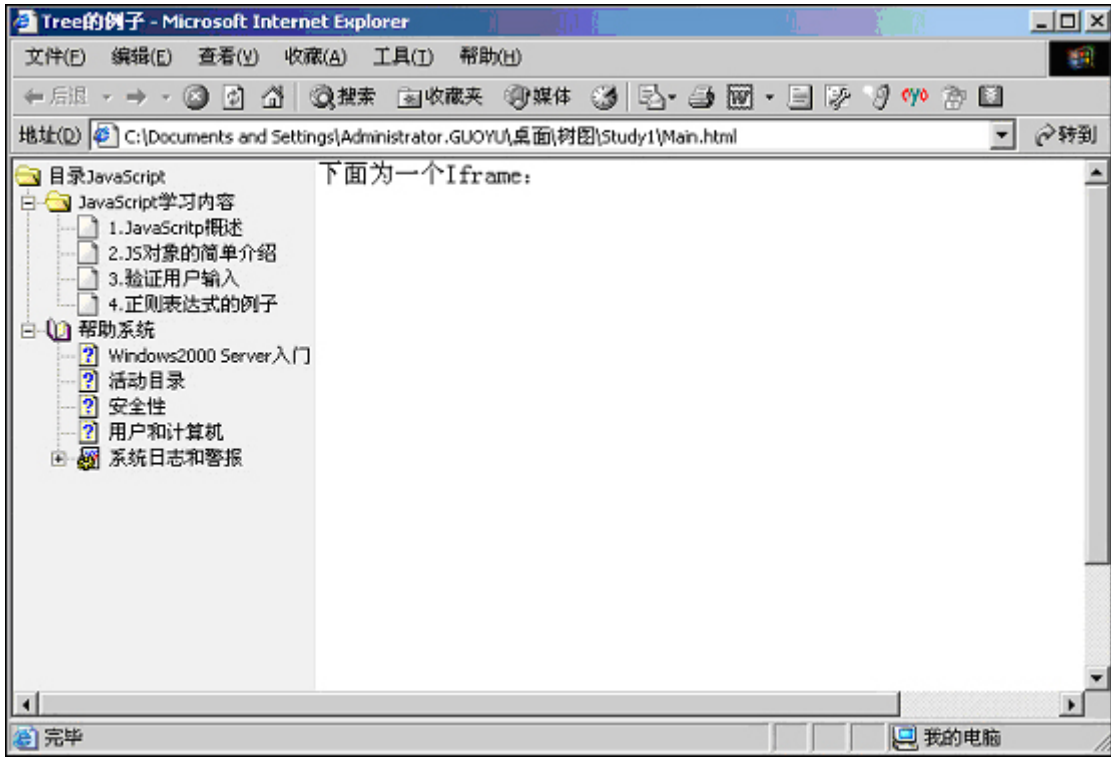
使用 Windows 操作系统的读者，大家都熟悉它的“资源管理器”对于目录和文件的内容组织，非常有助于用户浏览。下图是一个资源管理器的情况：

Figure 11: Windows 操作系统下的资源管理器



在 Web 应用程序中，也会存在很多的 Web 页面，如果将这些 Web 页面，灵活的组织在一起，以方便用户浏览？类似的，我们可以采用类似于上图的资源管理器的形式的 Tree 图。如下图：

Figure 12: 一个采用 Tree 图组织内容的网页



在 CD 上：

查阅随书光盘上的 JavaScript 目录下的“树图”目录下，Study0 目录下的内容是一些关于 Tree 的参考资料。我们主要根据 Study1 目录下的内容来讨论：

图 7 的 Main.htm 的源文件：

```
<html><head><title>Tree 的例子</title>
    <!-- 引入 Tree 的 Class 库文件-->
    <script src="xtree.js"></script>
    <!-- 引入 Tree 的样式表文件，修改该文件的内容可以实现 Tree 的一些外观风格的变化 -->
    <link type="text/css" rel="stylesheet" href="xtree.css">
    <style>body { background: white; color: black; }</style>
</head>
<body leftMargin=0 topMargin=0>
    <table border=0 cellspacing=0 height=100%>
        <tr valign=top><td bgcolor="#f1f1f1" width=190 >
        <!--Tree 的数据文件，利用数据来创建具体的 Class 的 Tree 的实例-->
        <script src="tree.js"></script>
        </td><td>下面为一个 Iframe：<br>
            <IFRAME name="ViewIframe" src="right.htm"
                    frameBorder=0 width=600 scrolling=auto height=400>
            </IFRAME>
        </td></tr>
    </table>
</body>
</html>
```

详细注释：

1. 同图 2-6 采用帧结构集相比，这里是采用在 Web 页上“开天窗”的方法（Iframe）。引入的 Iframe 的名称为 ViewIframe。
2. <script src="xtree.js"></script>，引入树图的 Class 文件。该文件中定义了树和它的分支的一些属性和方法。
3. <script src="tree.js"></script>，引入了树图的数据文件。它的源代码为：

```
if (document.getElementById) {
    //树的Class接口 ：WebFXTree(sText, sAction, sBehavior, sIcon, sOpenIcon)
    //分支的Class接口 ：WebFXTreeItem(sText, sAction, eParent, sIcon, sOpenIcon)

    var tree = new WebFXTree('目录JavaScript','');
    tree.setBehavior('classic');
    var a = new WebFXTreeItem('JavaScript学习内容');
    tree.add(a);
    a.add(new WebFXTreeItem('1.JavaScritp概述','Html/Js_1.htm'));
    a.add(new WebFXTreeItem('2.JS对象的简单介绍','Html/Js_2.htm'));
    a.add(new WebFXTreeItem('3.验证用户输入','Html/JsTest2.htm'));
    a.add(new WebFXTreeItem('4.正则表达式的例子','Html/JsTest5.htm'));

    var helpImg = "images/Helpfoldericon.gif";
    var OpenhelpImg = "images/openHelpfoldericon.gif";
    var b = new WebFXTreeItem('帮助系统','','',helpImg,OpenhelpImg);
    tree.add(b);
    var HelpfileImg = "images/Helpfile.gif";
    b.add(new WebFXTreeItem('Windows2000 Server入门','','',HelpfileImg ,HelpfileImg ));
    b.add(new WebFXTreeItem('活动目录','','',HelpfileImg,HelpfileImg));
    b.add(new WebFXTreeItem('安全性','','',HelpfileImg,HelpfileImg));
    b.add(new WebFXTreeItem('用户和计算机','','',HelpfileImg,HelpfileImg));

    var f = new WebFXTreeItem('系统日志和警报','','','images/Setfold.gif','images/Setfold.gif');
    b.add(f);
    f.add(new WebFXTreeItem('计数器日志','','','images/Setfile.gif','images/Setfile.gif'));
    f.add(new WebFXTreeItem('跟踪日志','','','images/Setfile.gif','images/Setfile.gif'));
    f.add(new WebFXTreeItem('警报日志','','','images/Setfile2.gif','images/Setfile2.gif'));
    document.write(tree);
}
```

在 xtree.js 文件中实现了树和分支的 Class 接口函数。
树的 Class 接口如下：WebFXTree(sText, sAction, sBehavior, sIcon, sOpenIcon)
分支的 Class 接口：WebFXTreeItem(sText, sAction, eParent, sIcon, sOpenIcon)

参数 sText,表示在网页上显示的值；sAction 表示链接的网页文件地址；sIcon 表示使用的图片，sOpenIcon 表示该分支含有子分支时，当展开该分支时显示的图片。在 xtree.js 文件的表示配置信息的结构 webFXTreeConfig 中，指出了当上面的某些参数没有指定时，所采用的默认值。从上面的代码的内容上看，有些参数没有指定，有些指定分支的所使用的图片和链接。

我们知道，对于链接来说，通过 Target 属性可以指定该链接的相应位置，从 Main.htm 的源文件来看，我们希望链接在名为 ViewIframe 的 Iframe 中显示，但 tree.js 库文件，没有指定 Target 属性的接口，所以我们需要在 xtree.js 文件中作一些修改：

请用<a href=\"" + this.action + "\" target=\"ViewIframe\"作为查找内容搜索 xtree.js 文件，这里的 target=\"ViewIframe\" 表示链接的 target 属性的设置，如果你的 Web 应用的 Iframe 的名称或者帧结构的名称不是 ViewIframe，请作相应修改。

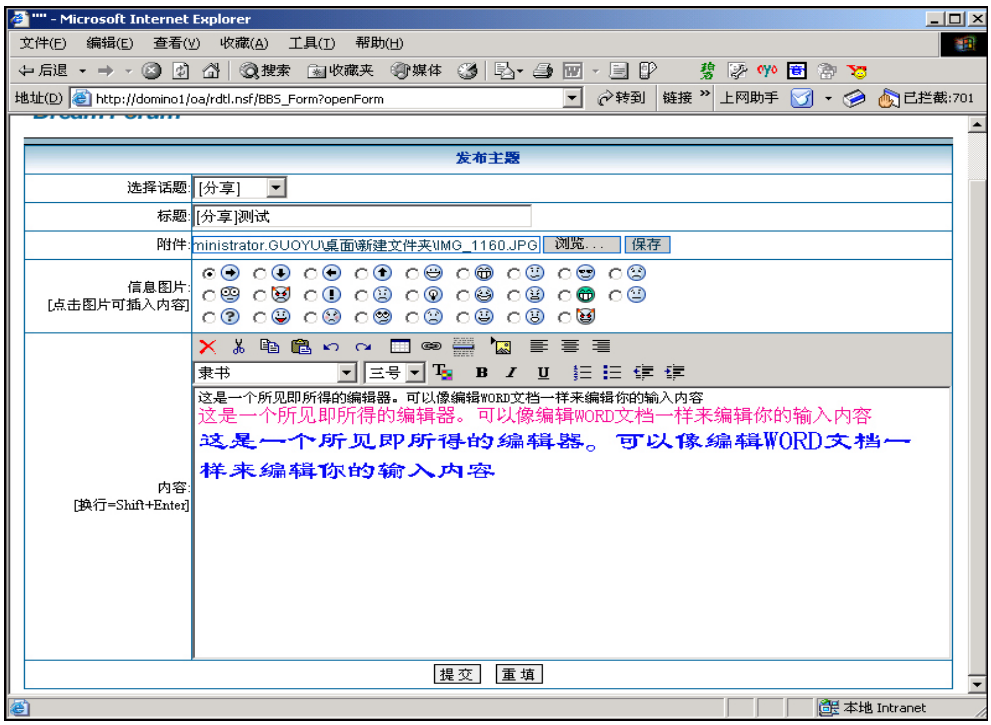
请读者参考光盘上的 JavaScript\树图\Study0 目录下的 Demo.htm 的例子，学习如何调用树的对象的某些方法。用户可以根据自己的需要可以对 xtree.js 和 tree.js 进行某些修改，比如从关系型数据库当中取得树和分支的数据信息，实现动态 Tree 的功能等。

从图 7 看来，这非常适合替代基于 Domino 的 Web 开发的“大纲”的作用。可自行定义的图片、并使用了高版本浏览器（计算机技术发展如此之快，希望你的浏览器的版本也要跟上时代的步伐，不要来自于古老年代的版本）所支持的 Persistence 特性（在这里的应用为为：当 Web 页刷新时，原来的 Tree 的分支的展开或者收缩状态保持不变）。这些特性比我们在 Web 设计时，采用 Domino 自身的大纲，要更加灵活、美观一些。关于 Persistence 的有关进一步的知识，请参考 www.msdn.com 的知识库文档。

5.6 一个很好的编辑器

为了在 WEB 应用程序中提供可以格式化的，更加丰富多彩的文本输入方式，很多论坛中采用了各种各样利用 JavaScript 代码编写的编辑器功能，让用户输入的内容能够通过 JavaScript 实现的按钮来产生 Html 的内容。在显示的时候，就能够提供丰富多彩的内容。如下图：

Figure 13: 所见即所得的编辑器的网页例子



在 CD 上：

查阅随书光盘上的 JavaScript 目录下的“编辑器”目录下，有该编辑器的例子和 JavaScript 代码。请仔细阅读。

6 总结和作业

6.1 总结

本文集中讨论了 JavaScript 的相关内容。在前半部分中说明了 JavaScript 的所访问的对象的基本概念—属性和方法，并提出了 JavaScript 访问对象的三个来源。接着讨论了用 JavaScript 访问浏览器环境和 Html 标签所构成的现成对象的主要方法---层层限定的逐步收缩法。并提供了几个典型的例子加以说明 JavaScript 的文档对象模型。框架窗口（FrameSet 和 IFrame）便于 Web 程序内容的组织，应用较为广泛。接着提到了在 JavaScript 使用正则表达式对用户输入内容进行模式匹配的有关内容。

在后半部分，提供了一些比较典型，较为复杂，应用范围较广的 JavaScript 的例子，这些优秀的代码可以拓展学习者的视野。通过阅读这些例子，有助于我们进一步加强对 JavaScript 的掌握。这些例子的选择，作者考虑到了基于 Domino 的 Web 程序设计的开发者可能会面临的问题，学习并掌握它们，对基于 Domino 的 Web 程序开发非常有帮助。

6.2 作业

这是技术文档。还不是像我们阅读小说作品一样，阅读完后只要能够得到心灵的愉悦感或者启迪就好，还不需要按照小说作品中的所描述的，自己去实践。但是技术文档的阅读，需要你按照文档的所提出的问题来实践，这样才能够真正的掌握。所以这里提供三个问题：

1．给 5.3 小节中“一个精美的日历”换衣裳。如图

请你参考 5.3 小节中提到的“精美的日历”的代码，作适当的修改，使之呈现上面的外观。通过这样的练习，可以让你进一步熟悉 HTML，CSS 和 JavaScript 内容。

2．让树图的功能更体贴。在 5.5 小节中给出了一个很优秀的树图的例子代码，现在我们假设我们要在这个树图提供的优秀的功能基础上，作进一步的扩展：做成“动态”的树图。树图的内容来自关系型数据库中，请你用服务器端 WEB 编程语言（ASP/PHP/JSP/PERL）读取树图的数据内容。然后传递给 WEB 页面文档的某个表单文本输入框对象（假定名称为 txtInput）中。仔细阅读树图的 JavaScript 代码，增加设计内容，实现解释表单对象 txtInput 的值，生成一个 Tree 对象。

3．让我们的视野更宽广。在 5.4 小节中提到的“下拉菜单”的例子。请到该例子的作者的网站上进一步了解作者提供的丰富多彩的外观设计例子。该“下拉菜单”很受国外的一些 WEB 编程者的欢迎。如果你上网冲浪，请作个有心人。也许你会发现其他的很多网站都是采用该作者的 JavaScript 代码来实现下拉菜单。但是外观风格却完全不一样。

This document has been created using MedBook V2.1

.