

以完成作品为教学导向，真正整合理论与实务

JavaScript

案例教程



- ▶ 精选数十个 JavaScript 实战案例，整理成十余种特效类型，悉心讲解其制作方法与使用技巧
- ▶ 使用者只需对案例中相关部分稍加修改，即可灵活套用到自己的网页中
- ▶ 既可供网页制作人员参考，又可作为培训教材

- ▶ 所有案例的页面效果与源代码
- ▶ 相关辅助工具



中科多媒体电子出版社

科海培中——全方位电脑培训服务提供商

案例教程系列

JSP 案例教程

ASP 案例教程

JAVA 案例教程

PHP 案例教程

Delphi 案例教程

Flash 5 案例教程

JavaScript 案例教程

Visual C++ 案例教程

Fireworks 4 案例教程

Visual Basic 案例教程

Director 8.5 案例教程

Authorware 6 案例教程

3DS MAX 4.0 案例教程

Photoshop 6.0 案例教程

动态网页设计案例教程

Dreamweaver 4 案例教程

Linux+PHP+MySQL 案例教程

Windows 2000+ASP+SQL Server 案例教程

ISBN 7-900084-16-9



9 787900 084163

封面制作：陆长景

出品：中科多媒体电子出版社

制作：北京科海培训中心

地址：北京海淀区中关村大街26号(100080)

技术电话：(010)82623337

销售电话：(010)62630320

网址：www.KHP.com.cn

定价：19.00元(1CD)

北京科海培训中心

JavaScript 案例教程

贾利峰 胡琳 编著

中科多媒体电子出版社

2001.9

内 容 提 要

JavaScript 是一种通用的、面向对象的高级脚本语言。本书精选了数十个 JavaScript 实例，并把它们整理成图像特效、鼠标特效、状态栏特效、页面特效和文本特效等十余种类型，一种类型组成一篇，每一篇又包含数个精彩的实例，分别从不同的方面讲述了某一种特效的制作方法和技巧。每一篇的内容既相互独立，又相互支持，全书浑然一体，集中介绍了 JavaScript 在网页制作上的各种技巧。

本书构思精巧，内容充实，实例新颖，语言风格大众化，是网络爱好者及网页制作人员理想的参考资料，也可作为相关人员的培训教材使用。本书的配套光盘中提供了所有案例的效果预览页面、源代码及相关辅助工具，详情请双击光盘根目录下的 `index.htm`。

品 名：JavaScript 案例教程

作 者：贾利峰 胡 琳

责任编辑：王超辉

出 版：中科多媒体电子出版社

印 刷 者：北京门头沟胶印厂

发 行：新华书店总店北京科技发行所

开 本：787×1092 1/16 印张：11.875 字数：274.5 千字

版 次：2001 年 9 月第 1 版 2001 年 9 月第 1 次印刷

印 数：0001~5000

盘 号：ISBN 7-900084-16-9

定 价：19.00 元（1CD）

前 言

随着 Internet 技术的突飞猛进, 各行各业都在加入 Internet 的行列中。而 WWW 发展到今天, 已成为当前 Internet 上最受欢迎、最为流行、最新型的信息资源。它利用超文本和超媒体技术结合 HyperLink (超链接) 的链接功能将各种信息组织成 Web (网络结构), 构成网络 Document (文档), 实现 Internet 上的“漫游”。而描述 WWW 网上信息资源的是 HTML (超文本标记语言), 通过 HTML 符号的描述就可以实现文字、表格、声音、图像、动画等多媒体信息的浏览。然而, 采用这种超链接技术存在一定的缺陷, 那就是它只能提供一种静态的信息资源, 缺少动态的客户端与服务器端的交互。虽然可通过 CGI (Common Gate Interface, 通用网关接口) 实现一定的交互, 但由于该方法编程较为复杂, 因而在一段时间内妨碍了 Internet 技术的发展。而 JavaScript 的出现, 有效地解决了 WWW 浏览所存在的问题, 这给 Internet 用户带来了一线生机。

JavaScript 的出现, 使得信息和用户之间不仅是一种显示和浏览的关系, 而且还具有了一种实时的、动态的、可交互的表达能力, 从而使基于 CGI 静态的 HTML 页面将要被这种可提供动态实时信息, 并对客户操作进行响应的 Web 页面所取代。JavaScript 脚本正是满足这种需求而产生的语言, 它深受广大用户的喜爱和欢迎, 是众多脚本语言中较为优秀的一种。因此, 尽快了解和掌握 JavaScript 脚本语言的基本知识和基本编程方法是广大用户的迫切要求。

现在, 有些读者积累的 JavaScript 知识可能已经足够丰富, 但是却很难把这些知识灵活地运用到实战当中。本书就是专门为这样的读者所写。本书没有采用传统教程的编写模式, 略去了基础知识部分, 而是用大量的案例来展示 JavaScript 在网页制作上的各种方法和技巧, 并配备有详尽的解释, 指导读者如何在这些案例中修改相关部分, 以便灵活地套用到自己的网页中。读完本书, 相信您对 JavaScript 的使用一定能更上一层楼。

本书精心整理了 JavaScript 的数十个案例, 并把它们归纳为图像特效、鼠标特效、状态栏特效、页面特效和文本特效等类型, 分别从各个不同的角度来揭示用 JavaScript 制作动态网页的方法和技巧。

由于编者水平有限, 不足之处在所难免, 恳请读者提出宝贵意见。

编者

2001 年 9 月

目 录

第 1 篇 图像特效	1
案例 1 改变图像隐现效果	1
案例 2 图片变形扭曲	3
案例 3 雪景	6
案例 4 相片选择器	11
案例 5 图像循环渐显	15
案例 6 图片响应鼠标变换	18
案例 7 图像浏览器	21
案例 8 水纹倒影	25
案例 9 图片自由运动	27
案例 10 飘动的图片	31
案例 11 图片虚幻表示	35
第 2 篇 鼠标特效	39
案例 12 鼠标经过打开新页面	39
案例 13 字符围绕鼠标	41
案例 14 追逐鼠标指针的图片	45
案例 15 跟着鼠标指针的字符	49
案例 16 鼠标跟踪器	53
第 3 篇 状态栏特效	57
案例 17 跳动的状态栏	57
案例 18 消失的状态栏信息	60
案例 19 “冒泡”的状态栏	62
案例 20 标题栏跑马灯	64
案例 21 状态栏跑马灯	66
案例 22 状态栏导航	68
案例 23 状态栏文字快速依次弹出	70
案例 24 状态栏文字组合弹出	74
第 4 篇 页面特效	79
案例 25 文档滚动特效	79
案例 26 改变背景颜色	81
案例 27 背景颜色连续变化	84
案例 28 时间决定背景颜色	87

案例 29 背景颜色表	90
第 5 篇 文本特效	94
案例 30 降落的文本	94
案例 31 缓缓上移的文本	98
案例 32 飘动的文本	101
案例 33 文字逐个闪耀	104
案例 34 旋转变换的文本	107
案例 35 文字效果变幻	110
案例 36 字符消隐特效	113
案例 37 文本自动输出特效	117
案例 38 文本颜色渐变	120
案例 39 文本弹跳特效	123
案例 40 元素周期表	126
第 6 篇 页面导航	129
案例 41 动态导航	129
案例 42 隐现导航	132
案例 43 下拉式导航菜单	137
案例 44 层叠式导航菜单	142
案例 45 目录式导航菜单	146
案例 46 移动导航菜单	149
案例 47 导航菜单说明	153
案例 48 自动变色的链接	157
案例 49 浮动链接导航条	159
案例 50 跑马灯式栏目指南	162
第 7 篇 其他特效	166
案例 51 设置打开窗口的特性	166
案例 52 日历	169
案例 53 追踪来访次数	173
案例 54 记录上次访问时间	178

第1篇 图像特效

案例1 改变图像隐现效果

【效果说明】

从本篇开始，将要讲述用 JavaScript 小程序处理图形图像的一些技巧，下面是一个最简单的例子。

打开本例的程序后，可以看到一幅模糊的图像，如图 1.1 所示。把鼠标移入图像区域，图像变得清晰了，如图 1.2 所示。而当鼠标移出后，图像又会恢复原样。

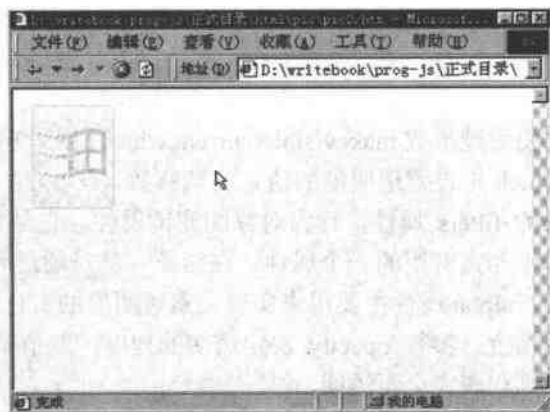


图 1.1 模糊的图像

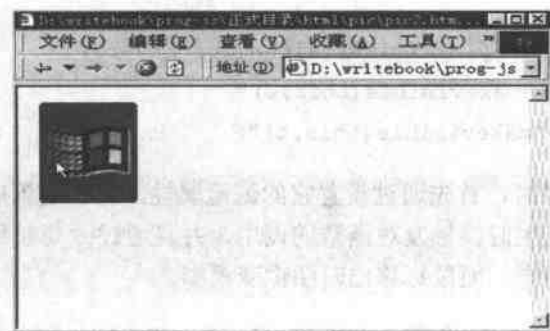


图 1.2 图像变清晰

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\pic\visible.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">

<!-- Begin
function makevisible(current,which){
if (which==0)
    current.filters.alpha.opacity=100
else
    current.filters.alpha.opacity=20
} // End -->

</script>
```

可以看到, 程序主要通过函数 makevisible(current,which)来改变图像对象的透明度, 并且通过传递来的参数 which 来设置透明度的值, 实现特效。

程序中用到的对象的 filters 属性, 称为对象的滤镜属性。它是页面特效制作, 尤其是图形图像方面特效制作中非常有用的一个属性。在这里, 就是通过它包含的一个 alpha 属性来设置图像的透明度。alpha 属性主要用来实现元素透明度的变化, 并可通过指定坐标, 来设置点、线、面的透明度。参数 opacity 决定透明的程度, 取值范围从 0~100, 其中 0 代表完全透明, 而 100 则代表完全不透明。

4. 将光盘内 sample\pic\visible.txt 中的“Part2”部分代码拷贝到<body>与</body>标记之间。

```

```

在页面中放置图像时, 首先通过设置它的滤镜属性, 将它设置成模糊的效果。然后当鼠标移入和移出事件发生时, 触发对函数的调用, 并且通过改变参数 which 的值来实现, 当鼠标移入时图像变清晰, 而鼠标移出时图像变模糊。

5. 将文件保存为 HTML 格式, 并在 IE (Internet Explorer) 中打开, 就可以看到前面所述的效果。

注意: 从本案例开始, 对案例中关于程序代码的解释段落加了底纹, 以便与一般文字有所区分。

案例2 图片变形扭曲

【效果说明】

本例也是一个很简单的使用 JavaScript 小程序实现图像特效的例子。

打开本例的程序后，可以看到一幅尺寸正常的图像，如图 2.1 所示。随着时间变化，图像沿 x 轴逐渐拉长、扭曲变形，如图 2.2 所示；然后又恢复原状，并如此不断往复循环。

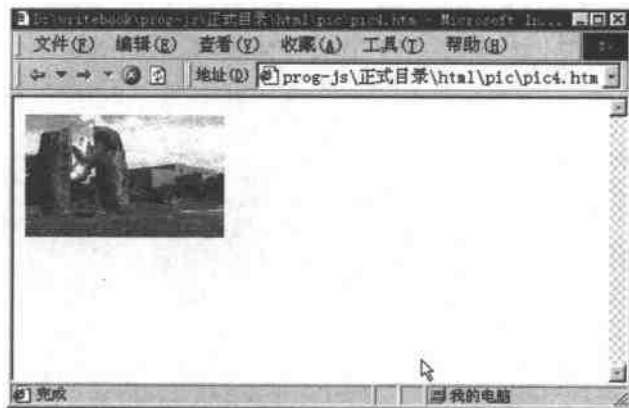


图 2.1 未变形的图像

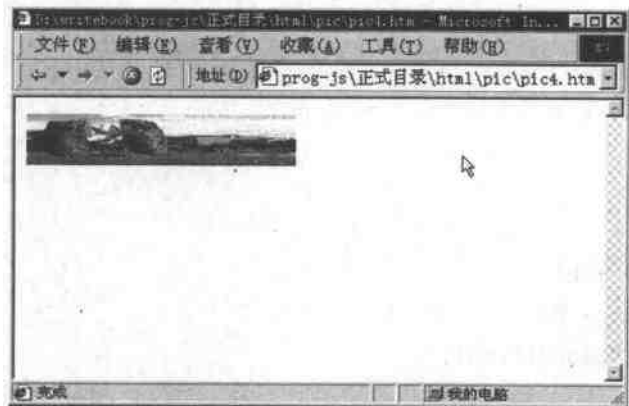


图 2.2 变形拉长的图像

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\pic\fade.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">

var b = 1;
var c = true;

function fade(){
    if(document.all){

        if(b==100) {
            c = false;
        }

        if(b==10) {
            c = true;
        }

        if(c == true) {
            b++;
        }

        if(c == false) {
            b--;
        }

        u.width=150 + b;
        u.height=125 - b;
        setTimeout("fade()",50);
    }
}</script>
```

程序代码一开始，先为两个变量设置初值 $b=1$, $c=true$ 。其中， b 的作用是改变图像的宽高的值，而 c 是用来决定 b 是否增减。

`function fade()` 为主要函数，用来实现动态改变图像的宽高值，其过程如下：

b 首先以 1 为单位递增；当 $b=100$ 时，则 $c=false$ ，此时， b 以单位 1 递减，当 $b=10$ 时，则 $c=true$ ，这时 b 又以单位 1 递增；

接着，函数把图像对象的宽度 `width` 和高度 `height` 对 b 分别做加减，就可以实现图像的扭曲变化。

最后，需要注意的是语句 `setTimeout("fade()",50)`，这个函数的功能是每隔第二个参数的时间就执行第一个参数所指的表达式，具体到这里来说，就是每隔 50 毫秒就调用函数 `fade()` 一次，如此可实现图像变形效果的往复循环。

4. 将光盘内 `sample\pic\visible.txt` 中的“Part2”部分代码拷贝到 `<body>` 与 `</body>` 标记之间。

```

```

在页面中只放置了一幅图像，并把它命名为 `u`，当然也可以用其他的名字来代替。不过要注意将前面程序中用到该名字的相应地方也要改正过来。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例3 雪 景

【效果说明】

相对前面的两个比较简单的例子来说，本例中介绍的这个“雪景”特效就有一定的难度了。

打开页面本例的程序后，可以看到片片雪花在窗口中飘飘落下，如图 3.1 所示，是不是很漂亮？而且，用户可以通过简单修改一下程序，使雪花变得“漫天飞舞”，如图 3.2 所示。很有意思吧！

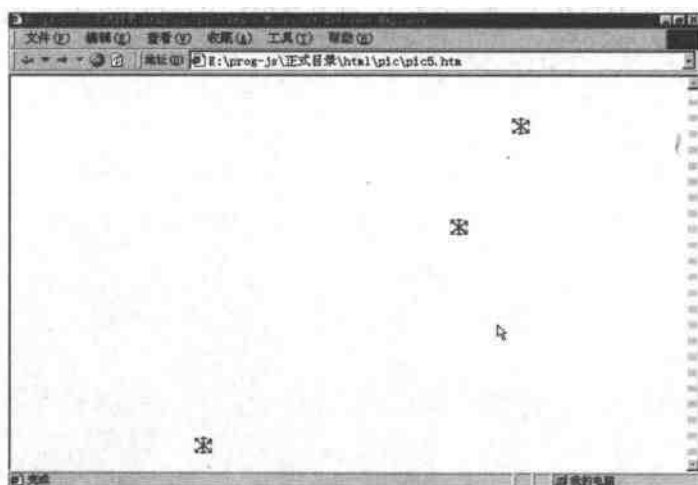


图 3.1 雪景 (1)

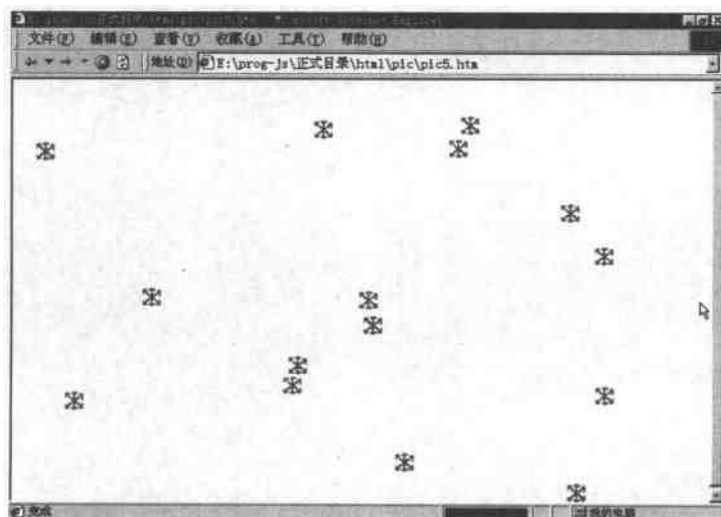


图 3.2 雪景 (2)

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\pic\snow.txt 中的“Part1”部分代码拷贝到<body>与</body>标记之间, 下面逐步讲解代码的作用。

先看看最开始的一段代码:

```
var no = 12; // number of hearts  
var heart = "heart.gif";
```

上述代码首先设定了雪花的数目, 用户可以在这里进行修改, 随自己的喜好选择雪花的数目。同时, 这里也选定了雪花的图片。当然, 用户可以选择自己喜欢的图片, 使之呈现雪花飘舞的效果。

注意: 把图片的路径设定好, 否则, 可能什么也看不到。

```
var flag;  
var ie4up = (document.all) ? 1 : 0;  
var dx, xp, yp; // coordinate and position variables  
var am, stx, sty; // amplitude and step variables  
var i, doc_width = 800, doc_height = 600;  
dx = new Array();  
xp = new Array();  
yp = new Array();  
amx = new Array();  
amy = new Array();  
stx = new Array();  
sty = new Array();  
flag = new Array();
```

上面的这一段代码设定了程序中要用到的一些变量, 并对它们进行了初始化。其中, ie4up 用来判断用户浏览器是否是 IE4.0 以上版本, 而 flag 是用来决定雪花飘动的左右方向的一个变量, dx、xp、yp 和 amx、amy、stx、sty 则分别是用来设定雪花位置和移动幅度等值的变量。显然, 每一片雪花都需要独自使用一组变量来描述, 所以需要用数组来保存这些变量。

```
if(ie4up){  
doc_width = document.body.clientWidth+document.body.scrollLeft;  
doc_height = document.body.clientHeight+document.body.scrollTop;  
}
```

```

for (i = 0; i < no; ++ i) {
    dx[i] = 0; // set coordinate variables
    xp[i] = Math.random()*(doc_width-30)+10; // set position variables
    yp[i] = Math.random()*doc_height;
    amy[i] = 12+ Math.random()*20; // set amplitude variables
    amx[i] = 10+ Math.random()*40;
    stx[i] = 0.02 + Math.random()/10; // set step variables
    sty[i] = 0.7 + Math.random(); // set step variables
    flag[i] = (Math.random()>0.5)?1:0;

    if (ie4up) {
        if (i == 0) {
            document.write("<div id=\"dot\"+ i +\"\" style=\"POSITION: ";
            document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
            document.write("visible; TOP: 15px; LEFT: 15px;\"><img src=\"");
            document.write(heart+ "\" border=\"0\"></div>");
        } else {
            document.write("<div id=\"dot\"+ i +\"\" style=\"POSITION: ");
            document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
            document.write("visible; TOP: 15px; LEFT: 15px;\"><img src=\"");
            document.write(heart+ "\" border=\"0\"></div>");
        }
    }
}
}
}

```

在上面的程序中，程序先获取了页面实际的宽度和高度：doc_width 和 doc_height，它们分别是用当前窗口的宽度 document.body.clientWidth 和高度 document.body.clientHeight 再加上因滚屏而造成的横向位移 document.body.scrollLeft 和纵向位移 document.body.scrollTop 而得到的。

然后用一个循环语句分别为每个雪花元素的变量赋初值，这里用到了一个数学方法 Math.random()，它可以产生一个 0~1 之间的伪随机值，这样，每片雪花的位置、移动、速度都是随机的。

最后，用标记<div></div>为每个雪花图片建立放置的容器，并为它们按序编号，指定路径和一些基本属性。

```

function snow() {
    doc_width = document.body.clientWidth+document.body.scrollLeft;
    doc_height = document.body.clientHeight+document.body.scrollTop;

    for (i = 0; i < no; ++ i) { // iterate for every dot

```

```
if (yp[i] > doc_height-50) {
    xp[i] = 10+ Math.random()*(doc_width-amx[i]-30);
    yp[i] = document.body.scrollTop;
    stx[i] = 0.02 + Math.random()/10;
    sty[i] = 0.7 + Math.random();
    flag[i]=(Math.random()<0.5)?1:0;
}

if (flag[i])
    dx[i] += stx[i];
else
    dx[i] -= stx[i];

if (Math.abs(dx[i]) > Math.PI) {
    yp[i]+=Math.abs(amy[i]*dx[i]);
    xp[i]+=amx[i]*dx[i];
    dx[i]=0;
    flag[i]=!flag[i];
}

document.all["dot"+i].style.pixelTop = yp[i] +
    amy[i]*(Math.abs(Math.sin(dx[i])+dx[i]));
document.all["dot"+i].style.pixelLeft = xp[i] +
    amx[i]*dx[i];
}
setTimeout("snow()", 10);
}
```

最后讲解一下主函数 snow()。

由于页面可能处于不停翻转状态中，因此，程序一开始就再次获取了变量 doc_width 和 doc_height，以备后面使用。

然后，以雪花数目“no”为循环上限，其目的是为了改变每个雪花的位置、移动方向等属性。开始作了一个判断“yp[i] > doc_height-50”，这是为了知道雪花是否已经飘到屏幕最下方。如果是，就把雪花的纵坐标值设到屏幕最上方的位置：yp[i] = document.body.scrollTop。

接下来，用变量 flag[i]来决定雪花飘动的方向是向左还是向右。在这里，程序通过一个判断“Math.abs(dx[i]) > Math.PI”：当 dx[i]的绝对值大于 π 之后，就作一次变向——将 dx[i]的值重新置 0，将 flag[i]取非。

做完各项工作之后，将每个雪花改变后的位置值赋给相应的保存有图片的容器，这样，

屏幕上的雪花才会真正改变位置。

最后，用函数 `setTimeout("snow()", 10)` 来每隔 10 毫秒调用主函数 `snow()`，实现循环。

4. 将 `<body>` 标记改成 `<body onload="snow()">`，以便在页面装载完毕时调用 `snow()` 函数。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例4 相片选择器

【效果说明】

在本例中，通过一个下拉列表框来控制图片的显示，它可以被用来制作相片保存簿，只要选中相应的文字就可以在窗口中看到相片了。

打开本例的程序后，可以看到一个“请选择相片”的下拉列表框，如图 4.1 所示。单击下三角按钮，可以看到选择相片的文字说明。任选中一个，如图 4.2 所示，会打开一个新窗口，并在其中显示所选的相片，如图 4.3 所示。单击新窗口中的“关闭窗口”按钮可以关闭它。

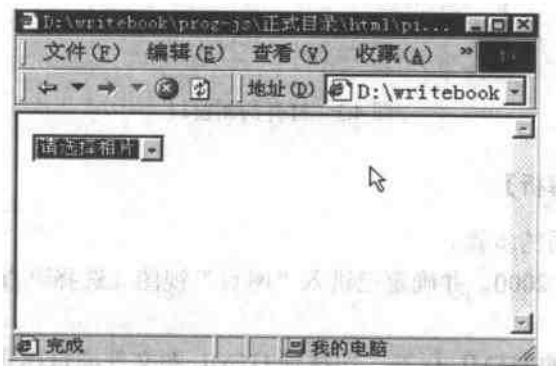


图 4.1 初始的窗口

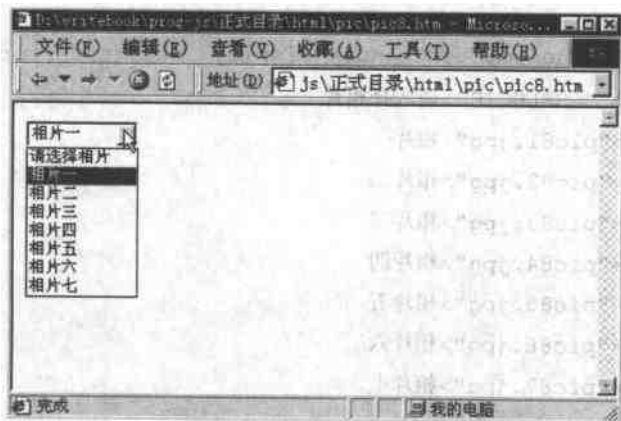


图 4.2 选择相片



图 4.3 打开的新窗口

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\pic\displayImage.txt 中的“Part2”部分代码拷贝到<body>与</body>标记之间。

```
<form>  
  <select NAME="imagename" onChange="display_image(this.form)">  
    <option value="" SELECTED>请选择相片  
    <option value="pic81.jpg">相片一  
    <option value="pic82.jpg">相片二  
    <option value="pic83.jpg">相片三  
    <option value="pic84.jpg">相片四  
    <option value="pic85.jpg">相片五  
    <option value="pic86.jpg">相片六  
    <option value="pic87.jpg">相片七  
  </select>  
</form>
```

上面一段代码的目的是制作一个包含一个下拉列表框的表单，用来向程序提交用户的选择，相信熟悉 HTML 的用户对此都很明白。需要注意的是，这里把此下拉列表框命名为 imagename，在后面的程序中可以通过这个名字来访问它。在发生 onChange（内容改变）事件时，调用函数 display_image()。

4. 将光盘内 sample\pic\displayImage.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">
  <!-- Begin
  function display_image(form) {
    selectionname=
      form.imagename.options[form.imagename.selectedIndex].text;
    selection= form.imagename.options[form.imagename.selectedIndex].value;

    Preview=window.open("", "Preview", "toolbar=0,location=0,directories=0,
    status=0,menubar=0,scrollbars=0,resizable=0,copyhistory=0,width=360,
    height=440 left=0 top=0");

    Preview.document.open();
    Preview.document.write("<html><head>");
    Preview.document.write("<title>Preview</title>");
    Preview.document.write("</head><body BGCOLOR=FFFFFF TEXT=000000>");
    Preview.document.write("<form><center><B><FONT SIZE=+1>" +
    selectionname + "</font></B><hr>");
    Preview.document.write("<img HSPACE=0 VSPACE=0 " +
    "SRC='" + selection + "'>");
    Preview.document.write("<hr><form><input type='button' VALUE='关闭窗口' "
    + "onClick='window.close()'></form>");
    Preview.document.write("</center>");
    Preview.document.write("</body></html>");
    Preview.document.close();
  }
  // End -->
</script>
```

上面一段程序是用来从提交的表单中获取相片的名字和对应的图像文件，并新建一个窗口来显示它，下面详细地解释一下。

一开始，程序就通过使用表单中下拉列表框 imagename 对应的变量数组 form.imagename.options 来获得下拉列表框中所包含的相片名字和相应的图像文件，变量 form.imagename.selectedIndex 为一个整数，它代表选择对象中选中选项的索引。而 text 和 value 这两个属性分别用来获取 option 标记指定的文本和控件值，在这里就是相片的名字和相应的图像文件。在获得这两个值以后把它们分别赋给 selectionname 和 selection 两个变量。剩下的工作就是新建窗口并显示相片了。其过程如下：

首先，程序新建一个窗口并把它赋给一个变量 Preview，window.open 方法用来建立窗

口，在这里把除了窗口高度和宽度以外的其余参数都设为 0。然后，通过使用 `Preview.document.open` 方法为这个窗口打开一个文档，并通过 `Preview.document.write` 方法向文档中写入内容，其实就是写一个 HTML 页面用来显示相片，并且用前面提到的变量 `selectionname` 和 `selection` 来引用相片文字和位置。最后，在页面完成之后调用 `Preview.document.close` 将文档清除。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例5 图像循环渐显

【效果说明】

本例与本篇的第一个例子有类似的地方，它们都是通过改变图像的透明度来实现特效。不同的是，第一个例子是通过鼠标来触发图像改变，而本例是自动使图像实现循环渐显。

打开本例的程序后，可以看到一张模糊的图片，效果如图 5.1 所示。渐渐地，图像变得清晰起来，如图 5.2 所示，并如此往复循环。



图 5.1 开始时模糊的图像



图 5.2 图像逐渐变得清晰

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\pic\opacity.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">
var b = 1;
var c = true;

function fade(){
if(document.all);
if(c == true) {
b++;

}

if(b==100) {
b--;
c = false
}

if(b==10) {
b++;
c = true;
}

if(c == false) {
b--;
}
u.filters.alpha.opacity=0 + b;
setTimeout("fade()",50);
}
</script>
```

上面列出了程序的主函数 `function fade()`, 通过对图片滤镜属性中透明度的改变来实现图像循环渐显的特效。由于在案例 1 中已经介绍过关于图像滤镜属性中透明度的知识, 这里就不再赘述, 下面我们主要来看看程序是如何实现让图像透明度循环变化的。

函数设置了两个变量 `b` 和 `c`, 其中 `b` 是用来设定图像的透明度值, 而 `c` 是决定 `b` 的值是增加还是减少: 初始时, `b=1`, `c` 的值为 `true`, `b` 逐一增加; 当 `b=100` 时, `c` 值变为 `false`, `b` 逐一递减; 当 `b=10` 时, `c` 的值重新为 `true`, `b` 逐一增加, 如此循环往复实现透明度变化。而后面的 `setTimeout("fade()",50)` (其作用的介绍见案例 2) 是实现循环的关键, 每隔 50 毫秒调用主函数一次。

4. 用光盘内 sample\pic\opacity.txt 中的“Part2”部分代码替换原来的<body>与</body>标记。

```
<body onload="fade()">  
  
</body>
```

这段代码的作用是，在页面装载完毕后触发对主函数的调用。图像初始透明度值设为 0，并为图像命名，在主函数中通过这个名字来访问图片。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例6 图片响应鼠标变换

【效果说明】

在本例中通过鼠标单击，实现页面中的图片随机变化。

打开本例的程序后，会显示一张图片，如图 6.1 所示；用鼠标单击这张图片，它就随机地改变成另一张图片，如图 6.2 所示。本例中只使用了两张图片，实际应用中可以使用更多的图片。



图 6.1 初始图片



图 6.2 鼠标单击后变换图片

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\pic\swap.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">
<!-- Begin
var rand1 = 0;
var useRand = 0;

images = new Array;
images[1] = new Image();
images[1].src = "image1.jpg";
images[2] = new Image();
images[2].src = "image2.jpg";
images[3] = new Image();
images[3].src = "image3.jpg";
images[4] = new Image();
images[4].src = "image4.jpg";

function swapPic() {
var imgnum = images.length - 1;
do {
var randnum = Math.random();
rand1 = Math.round((imgnum - 1) * randnum) + 1;
} while (rand1 == useRand);
useRand = rand1;
document.randimg.src = images[useRand].src;
}
// End -->
</script>
```

上面的代码包括程序的主函数和一些初始化过程, 下面我们就逐一讲解。

首先, 程序声明两个变量 rand1 和 useRand, 在后面它们分别被用来表示将要显示的图片位置和正在显示的图片位置。接着程序用一个数组来保存所需显示的图片文件名称, 在实际使用时, 用户可以根据需要改变文件名和数组长度。

然后, 在函数 function swapPic() 中, 处理如何随机取得与当前显示的图片不同的图片。函数先声明一个变量 imgnum, 它等于数组中图片的数目。这里需要注意的是, 由于没有使用数组中的 0 号变量, 所以 imgnum 等于数组长度 images.length 减去 1。

接着,函数在当 rand1 等于 useRand 时,也就是当将要显示的图片与当前显示的图片是同一个时,不断地改变 rand1 的值,以保证图片的变化。对 rand1 的变化是通过数学方法 Math.random 来实现的,rand1 的值应该是 1~4 之间的整数。随机数 randnum 的值在 0~1 之间,imignum-1 等于 3,则它们的乘积值范围为 0~3,经过方法 Math.round 四舍五入后变为 0~3 的整数,再加上 1,就成了 1~4 的整数。当 rand1 与 useRand 不同时,就把 rand1 的值赋给 useRand,然后改变页面中所显示的图片。

4. 用光盘内 sample\pic\swap.txt 中的“Part2”部分代码替换原来的<body>与</body>标记。

```
<body bgcolor="#fef4d9" onload="swapPic()">
<center>
<a onClick="swapPic();"></a><br>
<font face="Verdana" size="-2">click image to change</font>
</center>
</body>
```

在此段代码中,在页面装载完毕后触发对主函数的调用,使得每次打开页面时出现的图片都是随机的。图片被命名为 randimg,可以在函数中设置它的值。当图片发生鼠标单击事件 onClick 时,触发对函数的调用。

5. 将文件保存为 HTML 格式,并在 IE 浏览器中打开就可以看到前面所述的效果。

案例7 图像浏览器

【效果说明】

本例可以循环显示一组图片，它们之间通过淡入淡出来衔接。

当打开本例的程序后，显示了一张图片，如图 7.1 所示。渐渐地，图片逐渐与背景融合，变淡，如图 7.2 所示。当它彻底消失后，另一张图片又渐渐地显现出来，如图 7.3 所示。如此循环显示一组图片。

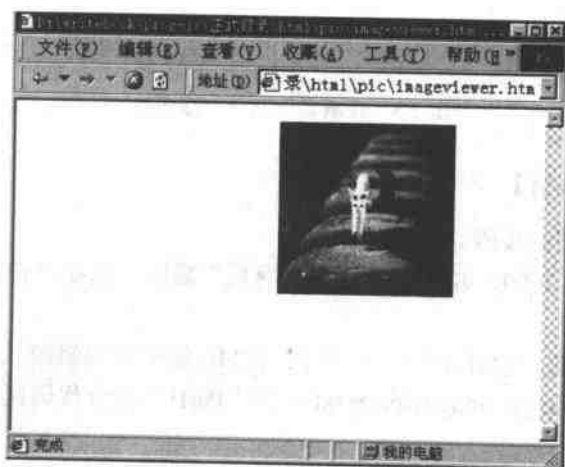


图 7.1 清楚显示的图片

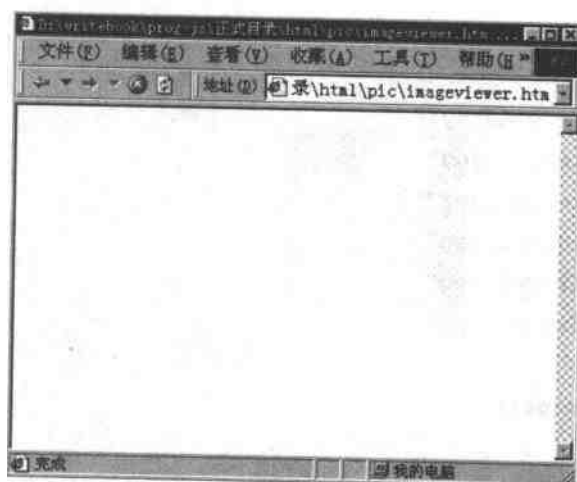


图 7.2 逐渐淡去的图片



图 7.3 逐渐显现的下一张图片

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\pic\imageviewer.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language=JavaScript>
<!--//
var i_strngth=1
var i_image=0
```

```
var imageurl = new Array()
imageurl[0] ="image1.jpg"
imageurl[1] ="image2.jpg"
imageurl[2] ="image3.jpg"
imageurl[3] ="image4.jpg"
imageurl[4] ="image5.jpg"
```

```
function showimage() {
if(document.all) {
if (i_strngth <=110) {
testimage.innerHTML="";
```

```
i_strngth=i_strngth+10;
var timer=setTimeout("showimage()",100)
}
else {
clearTimeout(timer)
var timer=setTimeout("hideimage()",10)
}
}
}

function hideimage() {
if (i_strngth >=-10) {
testimage.innerHTML="";

i_strngth=i_strngth-10
var timer=setTimeout("hideimage()",10)
}
else {
clearTimeout(timer)

i_image++
if (i_image >= imageurl.length) {i_image=0}

i_strngth=1
var timer=setTimeout("showimage()",200)
}
}
//-->
</script>
```

下面就来看看上面的一段程序是如何运作的。

首先,程序做了一些初始化工作,声明了两个变量 `i_strngth` 和 `i_image`,它们分别用来表示图片显示透明度的变化和所显示图片的编号,变量数组 `imageurl` 用来保存显示的文件的路径。

接着,程序的两个主函数 `showimage()` 和 `hideimage()` 分别实现图像渐显和淡出的工作。对 `showimage()` 来说,它首先在页面上放置图片的 `testimage` 容器中填入图片,并将它的透明度设置为 `i_strngth`。利用定时器 `timer=setTimeout("showimage()",100)` 触发对它的连续调用,每次 `i_strngth` 的值增加 10,实现图片的渐显。当 `i_strngth` 大于 110 后,程序触发对

hideimage()的调用。

与 showimage()类似的, hideimage()也是在该容器中填入图片, 透明度为 i_strngth, 并利用定时器来连续调用它。不同的是, 每次 i_strngth 的值减少 10, 实现图片的淡出。当 i_strngth 小于-10 后, 程序重新设定 i_strngth 和 i_image 的值: i_image 的值增加 1, 如果超过了数组的范围, 则设为 0; 而 i_strngth 的值重新设为 1。然后通过定时器触发对函数 showimage()的调用, 开始显示一张新的图片。

4. 用光盘内 sample\pic\imageviewer.txt 下的“Part2”部分代码替换原来的<body>与</body>标记。

```
<body onLoad="showimage()">
<center>
<div id="testimage" style="position:absolute;visibility:visible;"></div>
</center>
</body>
```

这里主要是定义了一个标识为 testimage 的<div>对象, 它被用来作为显示图片的容器。在页面装载完毕后触发对函数 showimage()的调用。

5. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例8 水纹倒影

【效果说明】

本例在原有图片的下方显示了一个具有波动效果的原图像的倒影，就好像它映在水面上的倒影一样，如图 8.1 所示。



图 8.1 水纹倒影效果

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 用光盘内 sample\pic\invert.txt 下的“Part2”部分代码替换原来的<body>与</body>标记。

```
<body onload=f1()>  
  
<br>  
  
</body>
```


上述代码中，首先在页面上放置了一张图片。然后，再在原图片的下方放置了一张相同的图片，并通过设置图片的滤镜属性，使下方的图片倒置并产生水纹效果。在这里介绍一下所用到的3种滤镜属性 wave、blur 和 flipv。

wave 使对象产生垂直波纹效果，其中所用的参数 strength 代表波的振幅大小；freq 代表波的频率，也就是在此对象上共产生多少组完整的波纹；phase 代表相位，就是正弦波的初始角度，其取值在 0~100 之间，如果取 25 则初始角度为 90°；lightstrength 用于增强波纹的光影效果，它的取值也在 0~100 之间。

顾名思义，blur 就是使对象产生模糊的效果，主要是使倒影看来更逼真，这里使用了缺省参数。

flipv 的作用是使对象倒置，不然就无法形成倒影了。

4. 将光盘内 sample\pic\invert.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language=JavaScript1.2>
function f1(){
mdiv.filters.wave.phase+=10
setTimeout("f1()",100);
}
</script>
```

以上代码中 f1() 函数的主要作用是定时触发，改变倒影的相位。这样一来，倒影就如同真的浮在水中一样，产生波动的感觉，很漂亮。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例9 图片自由运动

【效果说明】

本例可以实现图片在窗口内的自由移动图片：从初始位置（即窗口左上角）出发，如图 9.1 所示，当它到达窗口边界时，它会改变移动方向，如图 9.2 所示。这样，图片始终在窗口范围内运动。现在很多网页上都能看到与此类似的小程序。

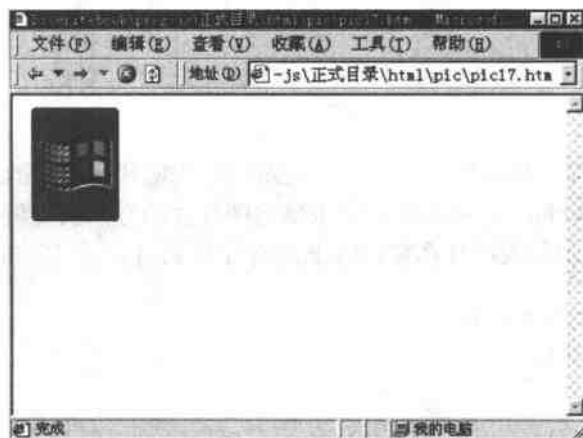


图 9.1 图片开始运动

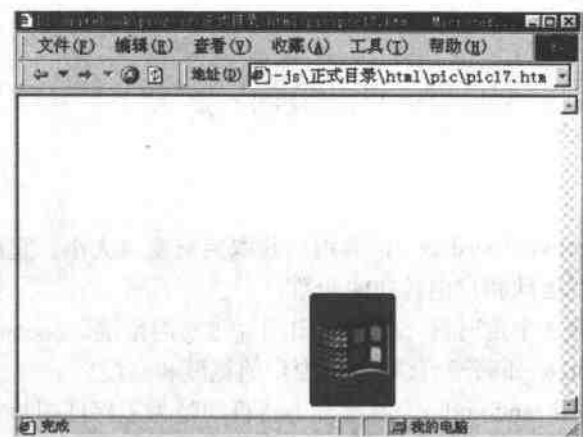


图 9.2 图片转向

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\pic\floating.txt 下的“Part1”部分代码拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">
<!-- Begin
var wwidth, wheight;
var ydir = '++';
var xdir = '++';
var id1, id2, id3;
var x = 1;
var y = 1;
```

上面一段代码是对后面所需用到的一些变量进行声明和初始化的工作, 这里包括表示窗口宽度和高度的 `wwidth` 和 `wheight`, 用来确定图片移动方向的变量 `xdir` 和 `ydir`, 3 个定时器 `id1`、`id2`、`id3` 以及代表图片在窗口内坐标位置的 `x`、`y`。

```
function getwindowsize() {
clearTimeout(id1);
clearTimeout(id2);
clearTimeout(id3);

wwidth = document.body.clientWidth - 55;
wheight = document.body.clientHeight - 50;

id3 = setTimeout('randomdir()', 20000);
animate();
}
```

上面的代码中, `getwindowsize()` 函数用来获取当前窗口大小。它同时也是整个程序的入口, 由它来调用实现运动和自由转向的函数。

一开始, 函数清除 3 个定时器, 接着求出图片运动的范围, `document.body.clientWidth` 和 `document.body.clientHeight` 分别代表当前窗口的宽度和高度。

函数定时 20 秒调用 `randomdir()` 函数, 主要目的是为了保证图片在一定时间以后才会改变运动的方向, 接着函数调用实现图片运动的函数 `animate()`。

```
function randomdir() {
if (Math.floor(Math.random()*2)) {
(Math.floor(Math.random()*2)) ? xdir='--': xdir='++';
} else {
(Math.floor(Math.random()*2)) ? ydir='--': ydir='++';
```

```

}
id2 = setTimeout('randomdir()', 20000);
}

```

前面已经提过, 函数 `randomdir()` 的作用是在图片运动一定时间以后改变它的运动方向, 而这种改变是随机的。表达式 `Math.floor(Math.random()*2)` 可能的取值是 0 或 1, 当值为 1 时, 函数改变图片在 X 轴上的运行方向; 为 0 时则改变图片在 Y 轴上的运行方向。至于具体的增减也通过表达式 `Math.floor(Math.random()*2)` 的取值来决定。

```

function animate() {
    eval('x'+xdir);
    eval('y'+ydir);

    pic1.pixelLeft = x+document.body.scrollLeft;
    pic1.pixelTop = y+document.body.scrollTop;

    if (pic1.pixelTop <= 5+document.body.scrollTop) ydir = '++';
    if (pic1.pixelTop >= wheight+document.body.scrollTop) ydir = '--';
    if (pic1.pixelLeft >= wwidth+document.body.scrollLeft) xdir = '--';
    if (pic1.pixelLeft <= 5+document.body.scrollLeft) xdir = '++';

    id1 = setTimeout('animate()', 30);
}
// End -->
</script>

```

函数 `animate()` 的作用是实现图片的具体运动, 同时它还在图片运动到窗口边缘时负责改变图片的运动方向。`eval('x'+xdir)` 和 `eval('y'+ydir)` 的作用是根据当前运动方向 `xdir` 和 `ydir` 来执行让坐标值 `x` 和 `y` 增加 1 还是减少 1 的操作。然后通过改变图片位置属性 `pic1.pixelLeft` 和 `pic1.pixelTop` 来实现图片的移位。接着, 函数判断图片是否到达窗口的 4 个边界, 如果是的话, 则改变图片移动的方向。例如, 当图片到达下边界时, `pic1.pixelTop >= wheight+document.body.scrollTop`, 则应该使运动方向朝上, 即 `ydir = '--'`, 这样, 当下一次计算坐标值的时候, `eval('y'+ydir)` 就会执行减 1 的操作。最后, 通过定时器 `id1` 每隔 30 毫秒调用函数一次, 实现动态效果。

4. 用光盘内 `sample\pic\floating.txt` 中的“Part2”部分代码替换原来的 `<body>` 与 `</body>` 标记。

```

<body bgcolor="#fef4d9" OnLoad="getwindowsize()"
        OnResize="getwindowsize()">
<div id="pic1" style="position:absolute; visibility:visible; left:0px;

```

```
        top:0px; z-index:-1">

</div>

<script language="javascript">
var pic1=eval('document.all.pic1.style');
</script>
</body>
```

此段代码中，首先使页面装载完毕和窗口改变大小时都调用函数 `getwindowsize()`。然后为图片设置容器，并标识为 `pic1`。再将此对象的 `style` 属性赋给一个对象变量 `pic1`，注意不要与前面的 `pic1` 混淆，这主要是为了方便在函数中实现对容器 `style` 属性的访问。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例10 飘动的图片

【效果说明】

本例与案例 9 有相似之处，也是图片在窗口内飘动。不过本例新增加了一个功能：用鼠标单击飘动的图片，图片就静止不动了，如图 10.1 所示；再单击一次，静止的图片又继续飘动，如图 10.2 所示。

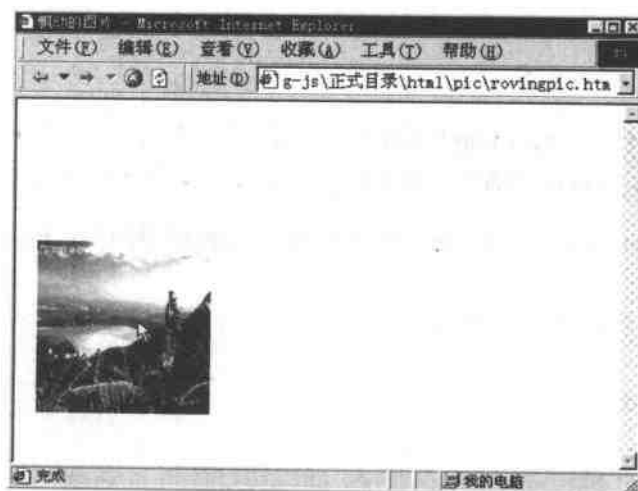


图 10.1 单击飘动的图片使它静止

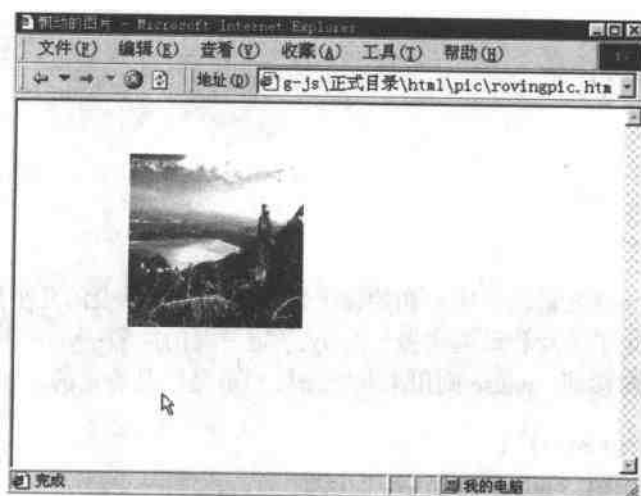


图 10.2 再单击它又继续飘动

【创作步骤与关键点解析】

下面是制作此页面的步骤。


1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。

3. 用光盘内 sample\pic\pause.txt 中的“Part2”部分代码替换原来的<body>!j</body>标记。

```
<body onLoad="start()">
<div id="img" style="position:absolute;">

</div>
</body>
```

上面的代码用于在标识为的<div>元素内放入一张图片, 并设置图片在发生鼠标点击事件时触发 pause_resume()函数, 在页面装载完成以后调用 start()函数。

4. 将光盘内 sample\pic\pause.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间, 下面逐段讲解。

```
<script language="JavaScript">
<!-- Begin
var xPos = 20;
var yPos = 20;
var step = 1;
var delay = 30;
var height = 0;
var Hoffset = 0;
var Woffset = 0;
var yon = 0;
var xon = 0;
var pause = true;
var interval;
```

程序首先进行一些变量的声明和初始化工作。其中比较重要的几个变量是: step 代表移动的步长, 它决定了图片移动的快慢; delay 是定时器的间隔; yon 和 xon 决定图片是沿坐标轴正向还是反向移动; pause 则用来判断图片当前是否是静止的。

```
function changePos() {
width = document.body.clientWidth;
height = document.body.clientHeight;
Hoffset = img.offsetTop;
Woffset = img.offsetLeft;
img.style.left = xPos + document.body.scrollLeft;
```

```
img.style.top = yPos + document.body.scrollTop;
if (yon) {
    yPos = yPos + step;
}
else {
    yPos = yPos - step;
}
if (yPos < 0) {
    yon = 1;
    yPos = 0;
}
if (yPos >= (height - Hoffset)) {
    yon = 0;
    yPos = (height - Hoffset);
}
if (xon) {
    xPos = xPos + step;
}
else {
    xPos = xPos - step;
}
if (xPos < 0) {
    xon = 1;
    xPos = 0;
}
if (xPos >= (width - Woffset)) {
    xon = 0;
    xPos = (width - Woffset);
}
}
```

函数 `changePos()` 的作用是实现在图片位置的改变, 形成飘动的效果。这里用到的方法跟上一例差不多, 读者可以比较这两个例子具体实现上的一些差异。需要注意的是, 本例在判断图片是否到达窗口边界时使用了 `img.offsetHeight` 属性, 这就使得当图片任何部分到达窗口边缘时, 图片就会转向, 这与上一例是有所区别的。而且本例可以通过调整 `step` (步长) 的大小来控制图片移动的快慢。

```
function start() {
    img.visibility = "visible";
    interval = setInterval('changePos()', delay);
}
```



```
}
```

以上代码中的 **start()** 函数是程序的入口，它实现对 **changePos()** 函数的定时调用。

```
function pause_resume() {  
  if(pause) {  
    clearInterval(interval);  
    pause = false;  
  }  
  else {  
    interval = setInterval('changePos()',delay);  
    pause = true;  
  }  
}  
-->  
</script>
```

pause_resume() 函数通过变量 **pause** 来判断当前图片是否在运动中，如果 **pause** 为真，则暂停图片的运动，并设 **pause** 为假；反之，则重新开始图片的运动，设 **pause** 为真。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例11 图片虚幻表示

【效果说明】

这是一个很有意思的例子：在页面上，一个圆形透镜在自由移动，透过它，可以隐约地看到下面的图片，如图 11.1 所示。

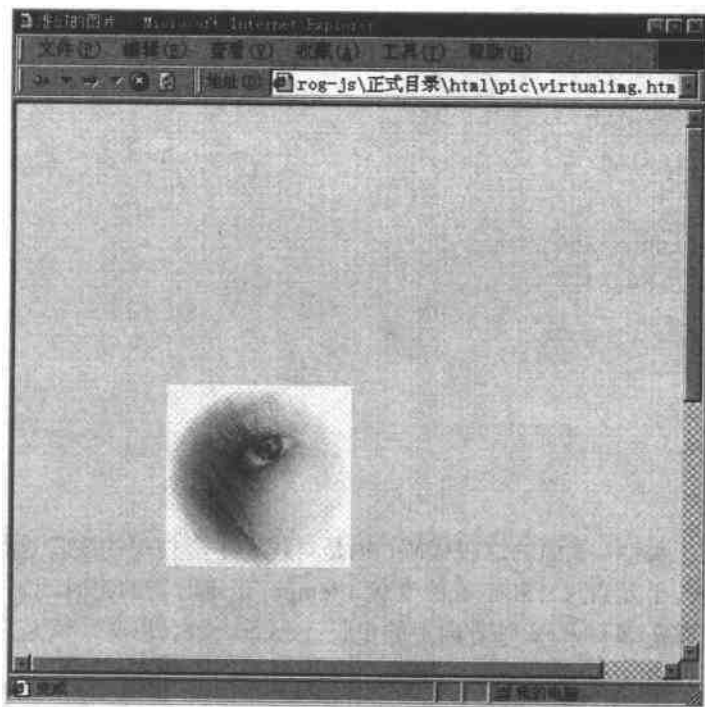


图 11.1 透过透镜看到虚幻的图像

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\pic\virtual.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间，下面开始逐段讲解。

```
<script language=JavaScript>
<!-- Beginning of JavaScript -
var x,y
var marginbottom
```

```
var marginleft=0
var margintop=0
var marginright

var cliptop
var clipbottom
var clipleft
var clipright
var clippoints

var ballheight=150
var ballwidth=150

var imageheight=525
var imagewidth=457

var tempo=25
var stepx=12
var stepy=6
var timer
```

同前面的例子类似，上面的这段代码，也是对程序中所使用的变量进行声明和初始化。其中的变量都是关于元素尺寸和定位的变量，**tempo** 是定时器的定时间隔；**stepx** 和 **stepy** 分别是透镜移动时在 X 轴和 Y 轴方向上的步长；**timer**，顾名思义，就是定时器。

```
function setValues() {
    if (document.all) {
        marginbottom = imageheight-ballheight
        marginright = imagewidth-ballwidth
        document.all.ball.style.posLeft=randommaker(400)
        document.all.ball.style.posTop=0
        document.all.textcontent.style.posLeft=0
        document.all.textcontent.style.posTop=0

        document.all.ball.style.filter="alpha(opacity=0,finishopacity=100,
            style=2,startX=0px,startY=0px,finishX=100px,finishY=100px)"
        moveball()
    }
}
```

在上述代码中, 函数 `setValues()` 首先找出透镜和图片边缘的界限, 这样, 在透镜移动到图片边缘时, 就可以改变它的移动方向。然后, 程序把图像和透镜的位置安排好, 它们是重叠在一起的, 其中透镜的 X 坐标通过调用函数 `randommaker()` 来得到, 这样透镜就可以出现在页面的任意位置。接着, 程序设置透镜的滤镜属性, 使位于图片上方的元素呈现圆形透镜效果。这里, `alpha` 属性的 `style` 属性设为 2, 代表透明度呈放射状, 且中心完全透明, 边缘完全不透明。效果起始和终止位置由 `startX`、`startY` 和 `finishX`、`finishY` 决定。最后, 函数调用另一个函数 `moveball()` 来使透镜移动。

```
function randommaker(range) {
    rand=Math.floor(range*Math.random())
    return rand
}
```

在上述代码中, 函数 `randommaker(range)` 用来求得一个范围为 0~range 的随机整数。

```
function moveball() {
    checkposition()
    if (document.all) {
        document.all.ball.style.posLeft+=stepx
        document.all.ball.style.posTop+=stepy

        cliptop=document.all.ball.style.posTop
        clipbottom=cliptop+ballheight
        clipleft=document.all.ball.style.posLeft
        clipright=clipleft+ballwidth

        clippoints="rect("+cliptop+" "+clipright+" "+clipbottom+"
                    "+clipleft+")"
        document.all.textcontent.style.clip=clippoints
        timer=setTimeout("moveball()",tempo)
    }
}
```

在上述代码中, 函数 `moveball()` 首先调用函数 `checkposition()`, 检查透镜的位置。然后移动透镜的位置, 此部分与前面两个例子的方法基本相同, 就不再详细讲解了。不过在透镜移动以后, 还要同步移动除透镜下方图片以外的部分, 使得在透镜下方的图片显现出来的同时, 遮挡其余的部分, 这一点需要注意。

```
function checkposition() {
    if (document.all) {
        if (document.all.ball.style.posLeft>=marginright) {
```

```

        stepx=stepx*-1
        document.all.ball.style.posLeft-=10
    }
    if (document.all.ball.style.posLeft<=marginleft) {
        stepx=stepx*-1
        document.all.ball.style.posLeft+=10
    }
    if (document.all.ball.style.posTop>=marginbottom) {
        stepy=stepy*-1
        document.all.ball.style.posTop-=10
    }
    if (document.all.ball.style.posTop<=margintop) {
        stepy=stepy*-1
        document.all.ball.style.posTop+=10
    }
}
}
// - End of JavaScript - -->
</script>

```

在上述代码中，函数 `checkposition()` 用于检查透镜的位置：当透镜到达下层图片边缘的时候，就改变它移动的方向。

```

<span id=textcontent style="LEFT: -5000px; position: absolute;
    top: -5000px"> </span>
<span id=ball style="background-color: white; height: 150px;
    position: absolute; top: -50px; width: 150px">
</span>

```

在上述代码中，放置了两个 `` 元素，分别用来装载图片和透镜，在程序中通过它们的标识 ID 来访问它们。

4. 用光盘内 `sample\pic\virtual.txt` 中的“Part2”部分代码替换原来的 `<body>` 和 `</body>` 标记。

```

<body bgcolor="#fef4d9" onload=setValues()></body>

```

页面中没有放置元素，只是作了底色的处理，并在页面装载完毕后调用函数 `setValues()`，来执行程序处理的部分。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

第2篇 鼠标特效

案例12 鼠标经过打开新页面

【效果说明】

本例是一个简单的鼠标特效程序。在浏览器中运行此程序，效果如图 12.1 所示，窗口中有一个红色的小箭头。

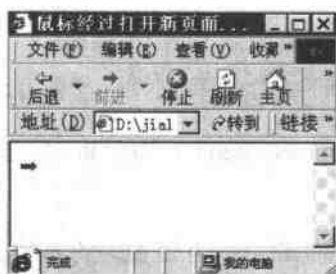


图 12.1 红色箭头

当鼠标指针移到这个红色的小箭头上时，就会打开一个新的文档，如图 12.2 所示。

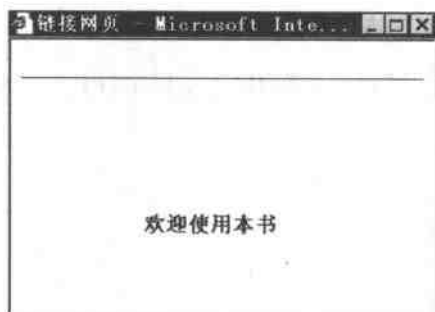


图 12.2 新的文档

【创作步骤与关键点解析】

下面是制作此例的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\mouse\mouseopen.txt 中的“Part1”部分代码拷贝到<head>与</head>标记之间, 下面逐段进行讲解。

程序首先创建了一个 winopen() 函数, 这个函数用来打开新的文档窗口, 如下所示。

```
<script language="JavaScript">
function winopen ()
{
    msg=open("blank.htm", "NewWindow", "toolbar=no, location=no,
            directories=no, status=no, menubar=no, scrollbars=no, resizable=
            no, copyhistory=yes, width=400, height=260");
}
</script>
```

第一个参数 blank.htm 代表新窗口所在的 URL 地址; 第二个参数 NewWindow 表示新窗口的名字; 第三个参数表示新窗口的各种属性, 读者可以自由更改这些设置。

4. 将光盘内 sample\mouse\mouseopen.txt 中的“Part2”部分代码拷贝到<body>与</body>标记之间。

程序在此部分创建了一个图形链接, 并设置它的 onMouseOver 事件驱动为 winopen() 函数。这样, 当鼠标移动到图形链接上时, 程序就执行 winopen() 函数, 从而打开新的文档。

```
<a href="" onMouseOver="winopen(); return true;">
<img SRC="shu2.gif" border="0">
</a>
```

5. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例13 字符围绕鼠标

【效果说明】

本例是一个有趣的鼠标特效程序。在浏览器中运行此程序，一串字符迅速变小，聚集在鼠标周围，快乐地跳舞，如图 13.1 和图 13.2 所示。



图 13.1 字符变小聚集



图 13.2 字符欢快地跳舞

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\mouse\txtcirc.txt 中全部内容拷贝到<head>与</head>标记之间，下面逐段讲解。

首先, 程序创建一个数组 **word**, 并存入 5 个字符, 代码如下:

```
var word=new Array(5);  
word[1]="J";word[2]="o";word[3]="y";word[4]="F";word[5]="m";
```

然后对这 5 个字符分别建立容器, 代码如下:

```
for(i=1;i<=5;i++)  
{  
    document.write("<div id='word"+i+"' "  
        style='width:20px;height:20px;position:absolute;font-size:1000; "  
        visibility:hidden'><font face='Forte' "  
        color='#00FF00'>"+word[i]+"</font></div>");  
}
```

接下来用函数 **locate()** 来确定鼠标的位置, 并把鼠标的坐标存放在变量 **cx**, **cy** 中, 代码如下:

```
var cx=0;  
var cy=0;  
function locate()  
{  
    cx=window.event.x;  
    cy=window.event.y;  
}
```

鼠标移动时, 触发 **onmousemove** 事件, 使程序重新定位鼠标, 代码如下:

```
document.onmousemove=locate;
```

函数 **show(i)** 用来改变字符的大小, 代码如下:

```
function show(i)  
{  
    var w=eval("word"+i);  
    with(w.style)  
    {  
        visibility="visible";  
        s=parseInt(fontSize);  
        if(s>=200)  
            s-=100;  
        else if(s>90&&s<=100)  
        {  
            s-=80;
```

```
        clearInterval(val);
        if(i<5)
            val=setInterval("show('+(i+1)+')",20);
    }
    fontSize=s;
}
}
```

函数 `follow(i)` 用来确定字符的位置。这里该字符的横坐标通过计算获得，而纵坐标则是随机获取，代码如下：

```
function follow(i)
{
    var x;
    if(i<4)
        x=cx-50+i*10;
    else
        x=cx-25+i*10;
    var y=cy-20+Math.floor(Math.random()*40);
    w=eval("word"+i);
    with(w.style)
    {
        left=x.toString()+"px";
        top=y.toString()+"px";
    }
}
```

`start()` 函数先调用 `show(i)` 函数使字符动态变小，然后调用 `follow(i)` 函数使字符围绕鼠标跳动，代码如下：

```
function start()
{
    for(i=1;i<=5;i++)
    {
        val=setInterval("show(1)",20);
        setInterval("follow('"+i+"')",100);
    }
}
```

注意：代码 “`val=setInterval("show(1)",20);`” 设置 `show(i)` 函数每隔 20 毫秒运行一次。当字符全部变小后，`show(i)` 函数的 “`clearInterval(val);`” 语句使 `val` 失效。

最后，调用 `start()` 函数使程序开始运行，代码如下：

```
<script language=JavaScript>
<!--
    start();
//-->
</script>
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例14 追逐鼠标指针的图片

【效果说明】

打开本例的程序后，几张大大小小的图片紧追着鼠标指针不放，鼠标指针走到哪儿，图片就跟到哪儿。如图 14.1 和图 14.2 所示。



图 14.1 追逐鼠标指针的图片 (1)

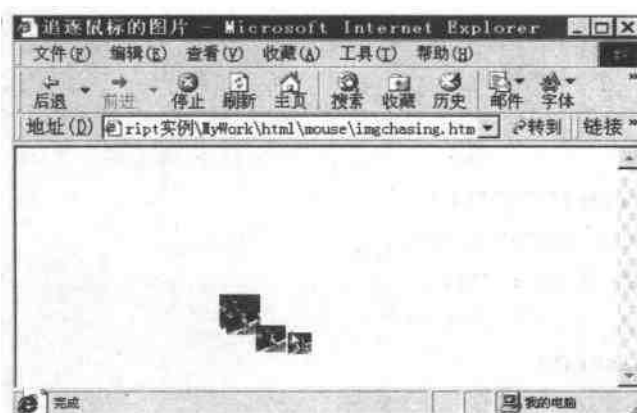


图 14.2 追逐鼠标指针的图片 (2)

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\mouse\imgchasing.txt 中的全部代码拷贝到<head>与</head>标记之间，下面逐段进行讲解。

变量 B 和 C 用来标识浏览器的版本。

```
B=document.all;  
C=document.layers;
```

函数 `createContainer()` 用来建立图像容器，其代码如下。其中：参数 N 为分配给各图像容器的 id；Xp 和 Yp 分别为图像的初始位置；W 和 H 为图像的宽度和高度。

```
function createContainer(N,Xp,Yp,W,H,At,HT,Op,St)  
{  
  with (document)  
  {  
    if(!B)  
      {write("<layer id='"+N+"' left='"+Xp+"' top='"+Yp+"' width='"+W+"' height='"+H);"  
      }  
    else  
      {write("<div id='"+N+"' style='position:absolute;left:"+Xp+";  
              top:"+Yp+"; width:"+W+"; height:"+H+";");"  
      }  
    if(St)  
    {  
      if (C)  
        {write(" style='");  
        write(St+";' ")  
        }  
      else write("{B}?"'':"");  
      write(({At}? At+">" : ">");  
      write(({HT} ? HT : "");  
      if (!Op)  
        closeContainer(N)  
    }  
  }  
}
```

`closeContainer()` 函数用于补充建立容器所必须的标识，代码如下：

```
function closeContainer()  
{  
  document.write(({B}? "</div>": "</layer>")  
}
```

然后程序调用函数 `createContainer()` 来建立 5 个图像容器，代码如下：

```
T1=new Array("shu7.jpg",38,35,"shu7.jpg",30,31,
             "shu7.jpg",28,26,"shu7.jpg",22,21,"shu7.jpg",16,16)
nos=parseInt(T1.length/3)
for (i=0;i<nos;i++)
{
    createContainer("CUR"+i,i*10,i*10,i*3+1,i*3+2,"","<img src='"+T1[i*3]+
        "' width='"+T1[(i*3+1)]+"' height='"+T1[(i*3+2)]+"' border=0>")
}
```

函数 **getXpos()** 和 **getYpos()** 用来获取对象的位置, 代码如下:

```
function getXpos(N)
{
    return (B) ? parseInt(B[N].style.left) : C[N].left
}
function getYpos(N)
{
    return (B) ? parseInt(B[N].style.top) : C[N].top
}
```

函数 **moveContainer()** 用来将对象移动到指定的位置, 代码如下:

```
function moveContainer(N,DX,DY)
{
    c=(B) ? B[N].style :C[N];
    c.left=DX;
    c.top=DY
}
```

函数 **cycle()** 则利用前面创建的函数将 5 个图像对象移动到新的位置, 代码如下:

```
function cycle()
{
    if (document.all&&window.print)
    {
        ie5fix1=document.body.scrollTop;
        ie5fix2=document.body.scrollLeft;
    }
    for (i=0;i<(nos-1);i++)
    {
        moveContainer("CUR"+i,getXpos("CUR"+(i+1)),getYpos("CUR"+(i+1)))
    }
}
```

```
}
```

函数 `newPos()` 用于将图像对象 `CUR4` 移动到鼠标所在的位置，代码如下：

```
function newPos(e)
{
    moveContainer("CUR"+(nos-1), (B)?event.clientX+ie5fix1:e.pageX+2,
                  (B)?event.clientY+ie5fix2:e.pageY+2)
}
```

鼠标移动时触发 `onmousemove` 事件，程序执行 `newPos()` 函数，代码如下：

```
document.onmousemove=newPos
```

设置每隔 10 毫秒 (`rate=10`) 运行一次 `cycle()` 函数，代码如下：

```
setInterval("cycle()",rate)
```

最后总结一下程序的执行过程：鼠标移动时，先运行 `newPos()` 函数，把图像对象 `CUR4` 移动到鼠标所在的位置；然后每隔 10 毫秒运行一次的 `cycle()` 函数，则是由 `getXpos()` 和 `getYpos()` 函数获取 `CUR4` 的新位置，把 `CUR0~CUR3` 这 4 个图像对象也移动到鼠标所在的位置，从而实现了程序所要达到的效果。

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例15 跟着鼠标指针的字符

【效果说明】

打开本例的程序后，一串字符紧跟着鼠标指针，鼠标指针移到哪儿，它们就跟到哪儿。图 15.1 为鼠标上下抖动时的效果。



图 15.1 鼠标抖动字符跟随

鼠标静止时，字符也逐渐排成一行，如图 15.2 所示。



图 15.2 鼠标静止字符列队

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\mouse\txtfollow.txt 中的“Part1”部分拷贝到<head>与</head>标记之间，下面逐段进行讲解。

程序首先建立了一个 CSS 样式表 spanstyle，这个样式表将用来设置字符串的显示属性，代码如下：

```
<style type="text/css">
    .spanstyle
    {
        COLOR: #ffd8ff; FONT-FAMILY: 宋体; FONT-SIZE: 10pt; POSITION: absolute;
        TOP: -50px; VISIBILITY: visible
    }
</style>
```

在 JavaScript 脚本部分，程序首先定义了要使用的字符串，并把它打散成一个个的字符，代码如下：

```
var message="★北京科海书店欢迎您的光临!"
message=message.split("")
```

然后程序创建数组 xpos 和 ypos 来存储各字符的位置，并给它们赋初值，代码如下：

```
var xpos=new Array()
for (i=0;i<=message.length-1;i++)
{
    xpos[i]=-50
}
var ypos=new Array()
for (i=0;i<=message.length-1;i++)
{
    ypos[i]=-200
}
```

接着用函数 handlerMM() 来确定鼠标的位置，代码如下：

```
function handlerMM(e)
{
```

```

x = (document.layers) ? e.pageX : document.body.scrollLeft+event.clientX
y = (document.layers) ? e.pageY : document.body.scrollTop+event.clientY
flag=1
}

```

最后，在函数 `makesnake()` 中，程序根据鼠标的位置来给各字符定位，代码如下：

```

function makesnake()
{
    if (flag==1 && document.all) {
        for (i=message.length-1; i>=1; i--) {
            xpos[i]=xpos[i-1]+step
            ypos[i]=ypos[i-1]
        }
        xpos[0]=x+step
        ypos[0]=y
        for (i=0; i<message.length-1; i++) {
            var thisspan = eval("span"+(i)+"."+style")
            thisspan.posLeft=xpos[i]
            thisspan.posTop=ypos[i]
        }
    }
    else if (flag==1 && document.layers)
    {
        for (i=message.length-1; i>=1; i--) {
            xpos[i]=xpos[i-1]+step
            ypos[i]=ypos[i-1]
        }
        xpos[0]=x+step
        ypos[0]=y
        for (i=0; i<message.length-1; i++) {
            var thisspan = eval("document.span"+i)
            thisspan.left=xpos[i]
            thisspan.top=ypos[i]
        }
    }
    var timer=setTimeout("makesnake()",30)
}

```

4. 将光盘内 `sample\mouse\txtfollow.txt` 中的“Part2”部分拷贝到 `<body>` 与 `</body>` 标记之间。

这里程序给每个字符分配了一个容器，代码如下：

```
for (i=0;i<=message.length-1;i++)
{
    document.write("<span id='span"+i+"' class='spanstyle'>")
    document.write(message[i])
    document.write("</span>")
}
```

鼠标移动时触发 **handlerMM** 函数，从而获取鼠标的新位置。代码如下：

```
document.onmousemove = handlerMM;
```

5. 将<body>语句换成如下代码：

```
<body bgcolor="#fef4d9" onload="makesnake()">
```

这样页面加载时，程序就执行 **makesnake()** 函数。

最后总结一下程序执行过程：页面加载时，程序运行 **makesnake()** 函数，对字符进行初始定位；鼠标移动时，运行 **handlerMM()** 函数，获得鼠标的新位置 (x, y)；每隔 30 毫秒运行一次的 **makesnake()** 函数，根据 x 和 y 的值将字符按顺序逐渐移动到鼠标指针附近。这样就实现了程序所要达到的效果。

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例16 鼠标跟踪器

【效果说明】

打开本例的程序后，几个星形的字符紧跟着鼠标指针，一步也不落下，如图 16.1 所示。



图 16.1 星形字符紧跟着鼠标指针

当鼠标指针停止移动时，这些字符都躲了起来，如图 16.2 所示。

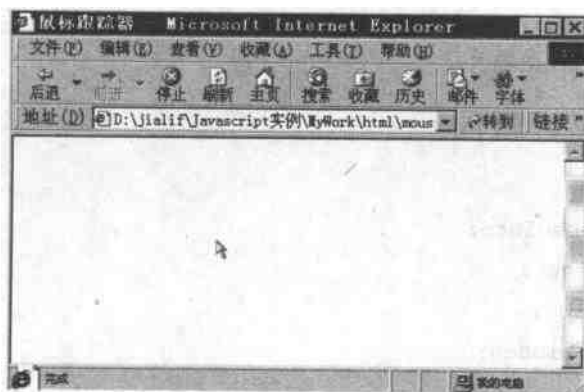


图 16.2 鼠标指针静止字符消失

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\mouse\mousechaser.txt 中的“Part1”部分拷贝到<head>与</head>标记之间，下面逐段进行讲解。

这一部分代码定义了4个CSS样式，用来确定显示星形字符的属性：

```
.s1
{
    position : absolute;
    font-size : 12pt;
    color     : blue;
    visibility: hidden;
}
.s2
{
    position : absolute;
    font-size : 20pt;
    color     : red;
    visibility : hidden;
}
.s3
{
    position : absolute;
    font-size : 16pt;
    color     : gold;
    visibility : hidden;
}
.s4
{
    position : absolute;
    font-size : 14pt;
    color     : lime;
    visibility : hidden;
}
```

4. 将光盘内 sample\mouse\mousechaser.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

利用前面定义的样式，程序为每个星形字符创建了一个容器。

```
<div id="div1" CLASS="s1">*</div>
<div id="div2" CLASS="s2">*</div>
<div id="div3" CLASS="s3">*</div>
<div id="div4" CLASS="s4">*</div>
```

函数 `get_mouse()` 用来确定鼠标指针的位置, 并调用 `beam()` 函数, 且参数为 1, 代码如下:

```
function get_mouse(e)
{
    x = (nav) ? e.pageX : event.clientX+document.body.scrollLeft;
    y = (nav) ? e.pageY : event.clientY+document.body.scrollTop;
    x += x_offset;
    y += y_offset;
    beam(1);
}
```

`beam()` 函数用于将星形字符显示在鼠标指针附近, 代码如下:

```
function beam(n)
{
    if(n<5)
    {
        if(nav)
        {
            eval("document.div"+n+".top="+y);
            eval("document.div"+n+".left="+x);
            eval("document.div"+n+".visibility='visible'");
        }
        else
        {
            eval("div"+n+".style.top="+y);
            eval("div"+n+".style.left="+x);
            eval("div"+n+".style.visibility='visible'");
        }
        n++;
        tmr=setTimeout("beam("+n+")", spd);
    }
    else
    {
        clearTimeout(tmr);
        fade(4);
    }
}
```

上段程序表示, 在进入 `beam()` 函数后, 函数先确定分区 `div1` 的位置, 然后将变量 `n` 的值加 1, 间隔 `spd` 毫秒后再执行 `beam()` 函数。这样就能确定 `div2` 到 `div4` 的位置。接着不再执行 `beam()` 函数, 转而执行 `fade(4)` 函数。

`fade()` 函数的作用是使星形字符消失, 其代码如下:

```
function fade(n)
{
    if (n>0)
    {
        if (nav)
            eval("document.div"+n+".visibility='hidden'");
        else
            eval("div"+n+".style.visibility='hidden'");
        n--;
        tmr=setTimeout("fade("+n+")",spd);
    }
    else clearTimeout(tmr);
}
```

鼠标指针移动时触发 `get_mouse` 函数, 从而获取鼠标指针的新位置, 其代码如下:

```
document.onmousemove = get_mouse;
```

5. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

第3篇 状态栏特效

案例17 跳动的状态栏

【效果说明】

本例是一个有趣的状态栏程序。在浏览器中运行此程序，浏览器的状态栏中就会出现跳跃的字符，如图 17.1 所示。

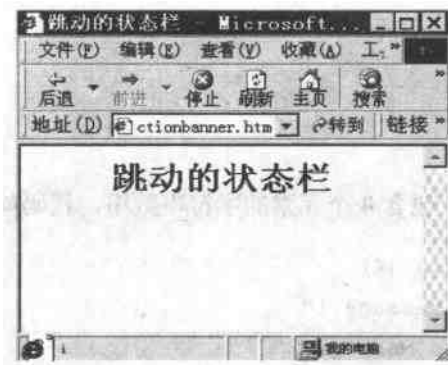


图 17.1 状态栏出现跳跃的字符

这些字符连接起来就组成一条完整的信息，如图 17.2 所示。



图 17.2 字符组成的信息

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\status\actionbar.txt 中的全部代码拷贝到<body>与</body>标记之间，下面逐段进行讲解。

在 JavaScript 脚本部分，程序首先创建了 `initArray(n)` 函数，这个函数用来建立数组，并将数组元素赋值为空的字符串。参数 `n` 就是要创建的数组元素的个数。

```
function initArray(n)
{
    this.length = n;
    for (var i =1; i <= n; i++)
    {
        this[i] = ' '
    }
}
```

然后，程序创建了一个包含 4 个元素的字符串数组，代码如下：

```
action = new initArray(4)
action[0]="This is Message 1"
action[1]="Now it is Message 2"
action[2]="No, do not say I have to do 3 messages"
action[3]="Yea, this message, 4, is the last message (and it's long)"
```

提示：读者如果想改变将要在状态栏中显示的信息，可以给 `action` 数组赋予其他任意的字符串，数组元素的个数也可以随意改变。

在接下来的 `action2()` 函数中，程序首先定义了两个变量：`current` 表示当前显示的字符串在 `action` 数组中的序号，`m` 则用于存储这个字符串。

```
function action2() {
    var current=0;
    var m=action[current]
```

接着程序用 `charAt()` 函数把字符串中的字符逐个显示在状态栏中，并让它停留 `speed` 毫秒。

```
window.status = m.charAt(x++)
setTimeout("action2()", speed)
```

当某个字符串中的字符全部显示一遍后，程序再将这个字符串整体显示一遍，代码如下：

```
if (x == m.length + 1)
{
    x = 0
    current++
    window.status = m
    ...
}
```

当 4 个字符串全部显示完一遍后，再重新开始，代码如下：

```
if (current > action.length - 1)
{
    current = 0
}
```

让字符串整体停留 `speed2` 毫秒后重新执行 `action2()` 函数：

```
setTimeout("action2()", speed2)
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例18 消失的状态栏信息

【效果说明】

本例是一个简单的状态栏程序。在浏览器中运行此程序，页面中就会以超链接的形式显示一条提示信息“请把鼠标放在这儿”。如图 18.1 所示。



图 18.1 提示信息

把鼠标指针放在这条提示信息上时，在状态栏中就会显示“这条消息很快就会消失！”的字符，如图 18.2 所示。然后信息很快消失。



图 18.2 出现状态栏信息

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\status\disbar.txt 中的全部代码拷贝到<head>与</head>标记之间，下面逐段进行讲解。

在 JavaScript 脚本部分，程序首先创建了 `moveover(txt)` 函数，这个函数用来显示状态栏信息。参数 `txt` 就是将要显示的字符串。函数先将参数 `txt` 赋给 `status` 对象，并设置 1000 毫秒（即 1 秒）后运行 `erase()` 函数。

```
function moveover(txt)
{
    window.status = txt;
    setTimeout("erase()",1000);
}
```

然后，程序创建了函数 `erase()`，将状态栏赋为空字符串，从而使状态栏信息消失。

```
function erase()
{
    window.status="";
}
```

接着程序创建了一个超链接，并将 `MouseOver` 事件的响应函数设为“`moveover('这条消息很快就会消失!')`”。这样，当鼠标指针放到超链接上时，就执行 `moveover()` 函数，使字符串“这条消息很快就会消失！”在状态栏显示出来，并在 1 秒后消失。

```
<a href="../../../blank/blank.htm" onMouseOver="moveover('这条消息很快就会消失!');
return true;">请把鼠标放在这儿.</a>
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例19 “冒泡”的状态栏

【效果说明】

本例是一个很有趣的状态栏特效程序。在浏览器中运行此程序，其状态栏就会显示出一条小写的英文信息，然后这些小写字母会一个个地变成大写字母。例如图 19.1 中“is”的“i”大写，图 19.2 所示中“bubble”的“b”大写。



图 19.1 “is”的“i”大写



图 19.2 “bubble”的第3个“b”大写

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\status\bubberbar.txt 中的全部代码拷贝到<body>与</body>标记之间，下面对重点部分进行讲解。

在 JavaScript 脚本部分，程序首先定义了状态栏要显示的字符串和显示的速度，代码如下：

```
var text = "this is the bubble banner"  
var speed = 200
```

将变量 **x** 设置为零，这个变量将决定哪些字符大写，哪些字符小写。

```
Var x=0
```

在接下来的 **bb()** 函数中，程序定义了 3 个字符串变量：

```
function bb(){  
var a = text.substring(0,x)  
var b = text.substring(x,x+1).toUpperCase()  
var c = text.substring(x+1,text.length)
```

上述代码用于将字符串 **text** 中 0~**x** 的字符赋给变量 **a**；将第 **x** 个字符的大写形式赋给变量 **b**；将 **x+1** 到末尾的字符赋给变量 **c**。

然后将字符串 **a**、**b**、**c** 连接起来赋给状态栏对象。代码如下：

```
window.status = a + b + c
```

再设置 **x** 的值，从零到最大，然后再从零开始。代码如下：

```
if(x == text.length)  
{  
    x = 0  
}  
else  
{  
    x++  
}
```

接着设置每隔 200 毫秒运行函数 **bb()** 一次。

```
Set Timeact("bb()",speed)
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例20 标题栏跑马灯

【效果说明】

本例是一个有趣的跑马灯程序，跑马灯通常出现在浏览器状态栏中，但是在本例中，跑马灯却出现在标题栏中。在浏览器中运行本程序，其标题栏中就会出现一条从左向右移动的消息，如图 20.1 所示。



图 20.1 标题栏的跑马灯

消息全部显示后，跑马灯结束，如图 20.2 所示。



图 20.2 禁止的跑马灯

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\\status\\titlepmd.txt 中的全部代码拷贝到<head>与</head>标记之间，下面对代码进行分析。

在 JavaScript 脚本部分，程序创建了如下变量：

index_count：显示字符的个数；

title_string：标题显示的字符串；

title_length: title_string 的总长度;

cmon: 启动跑马灯的事件;

kill_length: 跑马灯停止的时间。

```
var index_count = 0;
var title_string = "欢迎使用本书,这里有许多 JavaScript 源程序,
是您设计主页的好帮手! ";
var title_length = title_string.length;
var cmon;
var kill_length = 0;
```

在 scrollTheTitle() 函数中, 程序将 title_string 的某一部分赋给了字符串变量 doc_title。

```
Function ScrollTheTitle() {
    var doc_title = title_string.substring((title_length - index_count -
1),title_length);
```

然后将 doc_title 赋给对象 document.title, 使字符串以标题的形式显示出来。代码如下:

```
document.title = doc_title;
```

再使 index_count 增加 1, 从而使下次执行函数 scrollTheTitle() 时标题的长度增加了一个字符。代码如下:

```
index_count++;
```

在函数 loopTheScroll() 中, 首先执行 scrollTheTitle() 函数, 并设置每隔 100 毫秒执行一次 loopTheScroll()。代码如下:

```
scrollTheTitle();
cmon = setTimeout("loopTheScroll();",100)
```

执行完上述代码后, 将 kill_length 也加 1。当标题字符串 title_string 全部显示完后, 停止执行 LoopTheScroll() 函数:

```
kill_length++;
if(kill_length == title_length-1)
{
    clearTimeout(cmon);
}
```

这两个函数建立起来之后, 就可以调用函数 loopTheScroll() 执行程序了:

```
loopTheScroll();
```

4. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例21 状态栏跑马灯

【效果说明】

本例是一个简单的状态栏跑马灯。在浏览器中运行此程序，浏览器的状态栏中就会出现一条向左移动的消息，如图 21.1 所示。



图 21.1 状态栏跑马灯

消息全部显示后，跑马灯暂停 1 秒钟，然后重新开始运行，如图 21.2 所示。



图 21.2 跑马灯暂停 1 秒钟

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\status\statuspmd.txt 中的代码全部拷贝到<body>与</body>标记之间, 下面逐段进行讲解。

在 JavaScript 脚本部分, 程序首先定义了一个变量 myMsg, 它代表在状态栏中显示的字符串。代码如下:

```
myMsg = "希望这本书对您有所帮助! "  
i=0
```

在接下来的函数 scrollMsg() 中, 程序定义了两个字符串变量 frontPart 和 backPart: 将字符串 myMsg 的第 1 个到最后一个字符赋给 frontPart; 将字符串 myMsg 第 0 到第 i 个字符赋给 backPart。

```
frontPart = myMsg.substring(i,myMsg.length)  
backPart = myMsg.substring(0,i)
```

然后将这两个字符串拼起来赋给状态栏对象, 代码如下:

```
window.status = frontPart + backPart
```

接着控制变量 i 的值, 使它从 0 逐渐增加到 myMsg.length。当增加到 myMsg.length 时, 暂停 1 秒, 再运行 scrollMsg(); 否则, 每隔 200 毫秒运行一次 scrollMsg()。

```
if (i < myMsg.length)  
{  
    i++  
    setTimeout("scrollMsg()",200)  
}  
else  
{  
    i = 0  
    setTimeout("scrollMsg()",1000)  
}
```

4. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例22 状态栏导航

【效果说明】

本例是一个简单的状态栏导航程序，在浏览器中运行，其结果如图 22.1 所示。



图 22.1 程序运行初始状态

当鼠标指针移到某一个超链接上时，在状态栏中就会出现相应的导航信息，如图 22.2 所示。

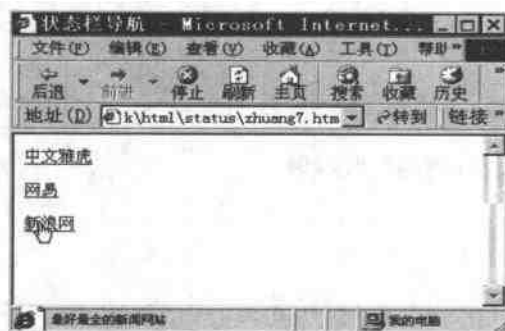


图 22.2 在状态栏显示导航信息

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\status\ztldh.txt 中“Part1”部分拷贝到<head>与</head>标记之间。

在本部分，程序定义了函数 MM_displayStatusMsg(msgStr)，用来在状态栏中显示字符串 msgStr。代码如下：

```
function MM_displayStatusMsg(msgStr)
{
    status=msgStr;
    document.MM_returnValue = true;
}
```

4. 将光盘内 sample\status\ztldh.txt 中“Part2”部分拷贝到<body>与</body>标记之间。

在本部分，程序创建了 3 个超链接“中文雅虎”、“网易”和“新浪网”。并将上面建立的 MM_displayStatusMsg() 设为 MouseOver 的事件驱动程序。将鼠标指针移到这 3 个超链接中任何一个上面时，程序就调用 MM_displayStatusMsg() 函数，将相应的字符串显示在状态栏中。

```
<a href="http://gbehinese.yahoo.com"
onMouseOver="MM_displayStatusMsg('中文雅虎搜索引擎!!!');
return document.MM_returnValue">
<font color="#0000FF">
中文雅虎
</font>
</a>
<br>
<br>
<a href="http://www.163.com"
onMouseOver="MM_displayStatusMsg('中国人气最旺的中文网站');
return document.MM_returnValue">
<font color="#0000FF">
网易
</font>
</a>
<br>
<br>
<a href="http://www.sina.com.cn"
onMouseOver="MM_displayStatusMsg('最好最全的新闻网站');
return document.MM_returnValue">
<font color="#0000FF">
新浪网
</font>
</a>
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例23 状态栏文字快速依次弹出

【效果说明】

本例是一个设计精巧的状态栏特效程序。在浏览器中运行此程序，浏览器状态栏中的字符依次先快后慢逐个向左移动。如图 23.1 和图 23.2 所示为“用”字向左移动的两个画面。



图 23.1 “用”字向左移动 (1)



图 23.2 “用”字向左移动 (2)

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\status\ycztl.txt 中的代码拷贝到<head>与</head>标记之间。

在 JavaScript 脚本部分，程序首先创建了函数 statusMessageObject()，这个函数在后面将被用来创建对象 scroll。

```
function statusMessageObject()  
{  
    this.msg = MESSAGE  
    this.out = " "  
    this.pos = POSITION  
    this.delay = DELAY  
    this.i = 0  
    this.reset = clearMessage  
}
```

其中, `clearMessage()` 函数由以下语句定义:

```
function clearMessage()  
{  
    this.pos = POSITION  
}
```

然后程序利用函数 `statusMessageObject()` 定义了对象 `scroll`:

```
var scroll = new statusMessageObject()
```

并且定义了如下 3 个变量:

```
var MESSAGE = "欢迎使用本书, 请多提意见。谢谢! "  
var POSITION = 100  
var DELAY = 4
```

其中 `MESSAGE` 表示将要在状态栏中显示的字符串; `POSITION` 表示字符串在状态栏中的初始位置; `DELAY` 控制字符移动的速度。结合函数 `statusMessageObject()`, 可以看出当前 `scroll` 对象的字符串:

```
scroll.msg = "欢迎使用本书, 请多提意见。谢谢! "  
scroll.out = " "  
scroll.pos = 100  
scroll.delay = 4  
scroll.i = 0  
scroll.reset = clearMessage
```

接下来的函数 `snapIn()` 是产生状态栏特效的主要函数。它的主要设计思想是用空格填充字符串, 并不断减少空格的个数。例如移动“用”字时, 先把“欢迎使”这三个字符赋给字符串 `out`, 然后填充一定数量的空格, 再将“用”字放在 `out` 的最后, 接着把 `out` 在状态栏中显示出来。然后不断减小填充的空格的个数, 这样在状态栏中观察到的效果就是“用”字不断向前(左)移动。为了达到先快后慢的移动效果, 每次移动(即减小)的空格数是

不相同的。在下面的程序解释中将会看到。

函数 `snapIn()` 有两个参数 `jumpSpaces` 和 `position`。`jumpSpaces` 就是状态栏显示字符中空格的个数；`position` 记录的是状态栏正在移动的字符在 `MESSAGE` 中的位置。程序开始运行时 `jumpSpaces` 的值为 100，`position` 的值为 0。

函数先将 `scroll.msg` 赋给字符串变量 `msg`，实际上就是将字符串变量 `MESSAGE` 的值赋给了 `msg`，代码如下：

```
var msg = scroll.msg
```

将状态栏中已经移动到左边的字符赋给 `out`。开始运行时 `position` 为 0，因而没有字符赋给 `out`，代码如下：

```
var out = ""
for (var i=0; i<position; i++)
{
    out += msg.charAt(i)
}
```

将空格赋在 `out` 后面，代码如下：

```
for (i=1; i<jumpSpaces; i++)
{
    out += " "
```

然后将正在移动的那个字符赋在最后面，代码如下：

```
out += msg.charAt(position)
```

将这个字符串在状态栏显示出来，代码如下：

```
window.status = out
```

下面来减小空格的个数。这里采用的方法是：

当空格的个数大于 3 时，将 $0.75 * \text{jumpSpaces}$ 赋给 `jumpSpaces`；当空格的个数小于等于 3 但是大于 1 时，将 `jumpSpaces` 减 1；当空格的个数小于或等于 1 时，将 `position` 加 1，改变要移动的字符；并将 “100-position” 赋给 `jumpSpaces`，为移动下一个字符作准备，代码如下：

```
if (jumpSpaces <= 1)
{
    position++;
    if (msg.charAt(position) == ' ')
    {
```

```
        position++
    }
    jumpSpaces = 100-position
}
else if (jumpSpaces > 3)
{
    jumpSpaces *= .75
}
else
{
    jumpSpaces--
}
```

当字符串变量 MESSAGE 的所有字符都移动到状态栏左边后（即 position 与 msg.length 相等），函数将 jumpSpaces 和 position 都赋为 0，并使程序重新开始运行；否则，程序暂停 4 毫秒后重新执行 snapIn() 函数。代码如下：

```
if (position != msg.length)
{
    var cmd = "snapIn(" + jumpSpaces + "," + position + ")";
    scrollID = window.setTimeout(cmd,scroll.delay);
}
else
{
    window.status=""
    jumpSpaces=0
    position=0
    cmd = "snapIn(" + jumpSpaces + "," + position + ")";
    scrollID = window.setTimeout(cmd,scroll.delay);
    return false
}
```

注意：由于这里设置 jumpSpaces=0，下一次重新运行时第一个字符“欢”字并不移动。

函数创建完成了，还需要调用它们才能运行：

```
snapIn(100,0);
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例24 状态栏文字组合弹出

【效果说明】

本例是一个奇特的状态栏跑马灯程序。在浏览器中运行此程序，浏览器状态栏中的字符就会随机分组并快速向左移动。如图 24.1 和图 24.2 所示。

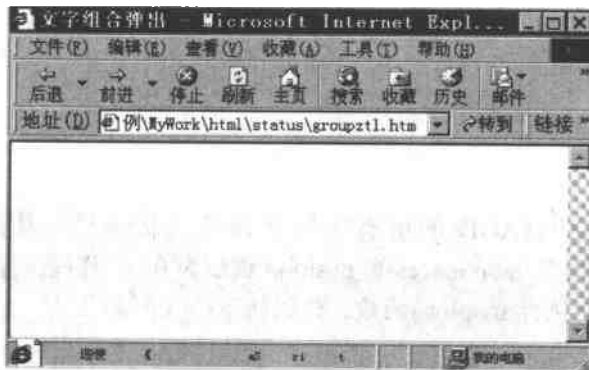


图 24.1 文字组合弹出 (1)



图 24.2 文字组合弹出 (2)

消息全部显示后，跑马灯暂停 1.5 秒钟，然后重新开始运行。但每次呈现的效果并不完全一样，因为这里采用了随机函数。

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\status\groupzt1.txt 的“Part1”部分拷贝到<head>与</head>标记之间。

下面先简单介绍一下这个程序的设计思想。程序先将字符串 `state` 赋为与状态栏字符串数组 `ar[n]` 相同长度的 0 字符，然后随机将某一位置上的 0 字符赋为 1。对应于字符串 `str`，字符串 `state` 中是 0 的位置，`str` 用若干空格表示；字符串 `state` 中是 1 的位置，`str` 就用 `ar[n]` 中相应位置的字符表示。这样，在状态栏中的效果就是字符串 `ar[n]` 中的字符分组显示出来。

程序的设计过程如下。

在 JavaScript 脚本部分，程序先定义了如下全局变量：

```
var speed = 10  
var pause = 1500  
var timerID = null  
var bannerRunning = false  
var message = 0  
var state = ""
```

各变量的含义如下：

speed：显示某一串字符时状态栏变化的时间间隔；

pause：显示完某一串字符后，状态栏暂停的时间；

timerID：某个函数对象的代号；

bannerRunning：控制状态栏特效的运行；

message：待处理的字符串在字符串数组 `ar[n]` 中的位置；

state：产生要在状态栏中显示的字符串。

然后程序创建了包含 3 个变量的字符串数组，代码如下：

```
var ar = new Array()  
ar[0] = "欢迎使用《JavaScript 案例教程》！"  
ar[1] = "错漏之处在所难免，请多包涵！"  
ar[2] = "请多提意见，谢谢！"
```

程序用函数 `clearState()` 对字符串变量 `state` 初始化，赋给它与状态栏字符串等长度的 0 字符。

```
function clearState()  
{  
    state = ""  
    for (var i = 0; i < ar[message].length; ++i)  
    {  
        state += "0"  
    }  
}
```

函数 `stopBanner()` 的作用是使状态栏特效停止，代码如下：

```
function stopBanner()
{
    if (bannerRunning)
    {
        clearTimeout(timerID)
        bannerRunning = false
    }
}
```

getRandom(max)函数的作用是得到一个最大值为 **max-1** 的随机数。代码如下：

```
function getRandom(max)
{
    return Math.round((max - 1) * Math.random())
}
```

这个函数先用 **Math.random()** 方法来产生一个 0~1 的随机数，与 **max-1** 相乘后，这个随机数的范围在 0 到 **max-1** 之间，然后利用 **Math.round()** 取整，得到与它最接近的整数值。这个函数在后面的 **getString()** 函数中用来产生 **state** 字符串。

下面来看一下 **getString()** 函数。这个函数利用前面介绍的 **getRandom()** 函数在已经赋零的 **state** 字符串中随机插入 1 而保持字符串的长度不变，也就是将随机产生的位置上的 0 赋成 1。

函数先定义了一个布尔变量 **full**，用来记录是否已经将 **state** 字符串中的 0 全部转变成了 1。它的初始值为 **true**。

```
var full = true
```

然后检查 **state** 中是否还有 0 存在，如果有，就将 **full** 赋为 **false**，代码如下：

```
for (var j = 0; j < state.length; ++j)
{
    if (state.charAt(j) == 0)
    {
        full = false
    }
}
```

如果 0 全部转变成了 1，则直接返回 **true**，代码如下：

```
if (full)
    return true
```

否则，随机找到 **state** 中某一为 0 的位置，代码如下：

```

while (1)
{
    var num = getRandom(ar[message].length)
    if (state.charAt(num) == "0")
        break
}

```

如果随机产生的位置的字符为 1，程序将重新产生随机数，直到找到一个数使 state 在该位置为 0 字符为止。然后把该位置上的 0 赋为 1，代码如下：

```

state = state.substring(0, num) + "1" + state.substring(num+1, state.length)
return false

```

下面来看最重要的 showBanner() 函数。

这个函数首先通过 getString() 函数得到包含字符 1 的字符串 state。由前面函数 getString() 的介绍可知，当 state 中全是 1 字符时，getString() 函数将返回 true。此时，字符串 ar[message] 处理完毕，将 message 加 1，转而处理下一个字符串。如果这 3 个字符串都已处理完毕，则将 message 赋为 0，从 0 开始重新处理，代码如下：

```

if (getString())
{
    message++
    if (ar.length <= message)
        message = 0
    clearState()
    timerID = setTimeout("showBanner()", pause)
    bannerRunning = true
}

```

当 state 中不全是 1 字符时，对应于字符串 str，字符串 state 中是 0 的位置，str 用若干空格表示；字符串 state 中是 1 的位置，str 就用 ar[message] 中相应位置的字符表示，代码如下：

```

var str = ""
for (var j = 0; j < state.length; ++j)
{
    str += (state.charAt(j) == "1") ? ar[message].charAt(j) : "   "
}

```

将字符串 str 在状态栏中显示出来，代码如下：

```

window.status = str

```

暂停 speed 毫秒后重新执行 showBanner()，代码如下：

```
timerID = setTimeout("showBanner()", speed)
```

将 bannerRunning 赋为 true，为下次运行做准备，代码如下：

```
bannerRunning = true
```

4. 将<body>标记换成如下代码，使页面登录时即运行 startBanner()函数：

```
<body bgcolor="#fef4d9" onLoad="startBanner()">
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

第4篇 页面特效

案例25 文档滚动特效

【效果说明】

本例是一个有趣的文档特效程序。在浏览器中运行此程序，浏览器窗口会不停地向下滚动，即文档向上滚动。如图 25.1 和图 25.2 所示。

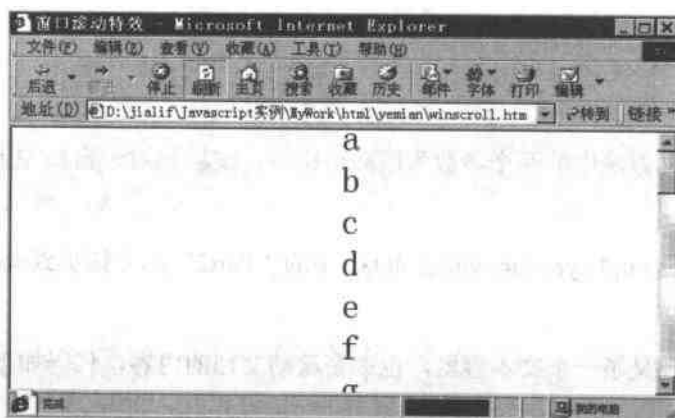


图 25.1 文档向上滚动 (1)

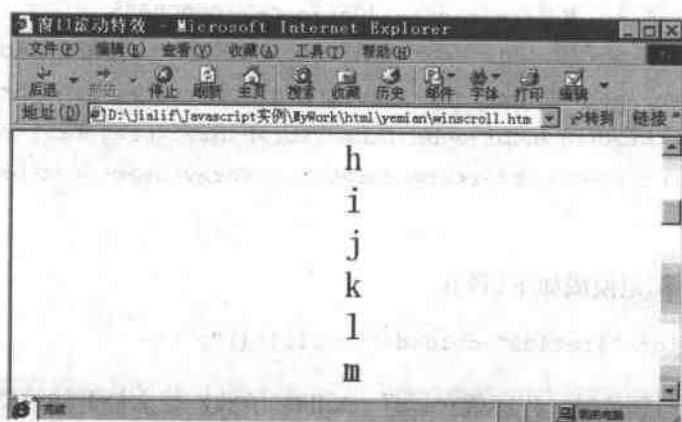


图 25.2 文档向上滚动 (2)

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\yemian\winscroll.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

在 JavaScript 脚本部分，函数 scrolllit() 的作用就是使文档滚动，代码如下：

```
function scrolllit()  
{  
    for (I=1; I<=500; I++)  
    {  
        parent.scroll(0,I)  
    }  
}
```

其中，scroll()方法中的两个参数为滚动结束后，文档相对于窗口左上角的横坐标和纵坐标。

4. 将光盘内 sample\yemian\winscroll.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

在这里程序定义了一个文本容器，也就是滚动文档的内容，代码如下：

```
<div id="glowdiv" style="position:absolute;visibility:visible;  
    width:600px;text-align:center; top:185px;left:70px;  
    font-family:隶书;font-size:30pt;color:000000">  
a<br>b<br>c<br>d<br>e<br>f<br>g<br>h<br>i<br>j<br>k<br>l<br>m<br>n  
<br>o<br>p<br>q<br>r<br>s<br>t<br>u<br>v<br>w<br>x<br>y<br>z<br>  
a<br>b<br>c<br>d<br>e<br>f<br>g<br>h<br>i<br>j<br>k<br>l<br>m<br>n  
<br>o<br>p<br>q<br>r<br>s<br>t<br>u<br>v<br>w<br>x<br>y<br>z  
</div>
```

5. 将<body>标记换成如下语句：

```
<body bgcolor="#fef4d9" onload="scrollit()">
```

这样，当文档登录时，程序就会运行 scrollit() 函数，使文档开始滚动。

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例26 改变背景颜色

【效果说明】

本例中的页面特效程序可以伪随机改变背景和字符的颜色。如图 26.1 所示，背景为深绿色，文本为浅绿色。



图 26.1 背景深绿文本浅绿

单击链接，背景变为黑色，而文本则变为绿色。如图 26.2 所示。



图 26.2 背景黑色文本绿色

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\yemian\changebgcolor.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

在此部分代码中，程序首先建立了一个超链接，代码如下：

```
<center>
<a href="javascript:history.go(0)">点击这里得到新颜色</a>
</center>
```

单击这个链接，程序就会执行 history.go(0) 方法，实际上就是把文档重新加载一遍。然后程序建立了两个文本框，分别用来显示当前使用的背景颜色和文本颜色，代码如下：

```
<form name=blah>
<table border=0 width=100%><tr><td align=left>文本颜色</td>
<td align=right>背景颜色</td>
</tr>
<tr>
<td align=left>
<input type=text name=fg>
</td>
<td align=right>
<input type=text name=bg>
</td>
</tr>
</table>
</form>
```

4. 将光盘内 sample\yemian\changebgcolor.txt 中的“Part2”部分拷贝到</Html>之后。

在此部分脚本中，程序首先创建了一个伪随机函数 rnd()，此函数用来产生一个伪随机数，代码如下：

```
function rnd(scale)
{
    var dd=new Date();
    return ((Math.round(Math.abs(Math.sin(dd.getTime())))*1000000000)%scale)
};
}
```

之所以说它是伪随机数，是因为它先创建一个时间变量 dd，然后得到当前的时间值（从 1970 年 1 月 1 日零时零分零秒记起的毫秒值），取它的正弦值乘以 1000000000 后依次取绝对值和取整，再对 scale 求余数，经过此番转换之后，谁都无法预料最后得到的数是多少。

但它并不是通过 `random()` 方法得到的随机数，所以只能说得到的是一个伪随机数。

然后程序通过这个函数来获得背景颜色和文本颜色，代码如下：

```
document.fgColor=256*rnd(255)+16*rnd(125)+rnd(255);  
document.bgColor=256*rnd(125)+16*rnd(255)+rnd(125);
```

函数 `show()` 用于把背景颜色值和文本颜色值显示在前面建立的文本框内，代码如下：

```
function show()  
{  
    document.blah.bg.value=document.bgColor;  
    document.blah.fg.value=document.fgColor;  
}  
show();
```

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例27 背景颜色连续变化

【效果说明】

本例是一个背景颜色连续变化的页面特效程序。在浏览器中运行此程序，“背景颜色开始变化咯”按钮后，背景颜色就会不停地改变。

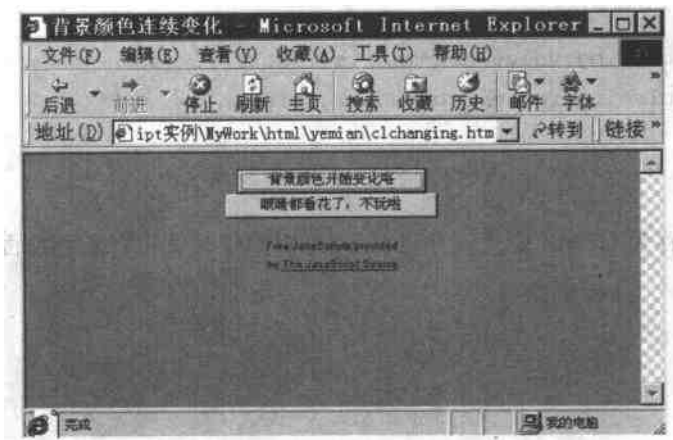


图 27.1 背景颜色变为灰色

当单击“眼睛都看花了，不玩啦”按钮后，背景颜色变为黑色后停止变化。如图 27.2 所示。

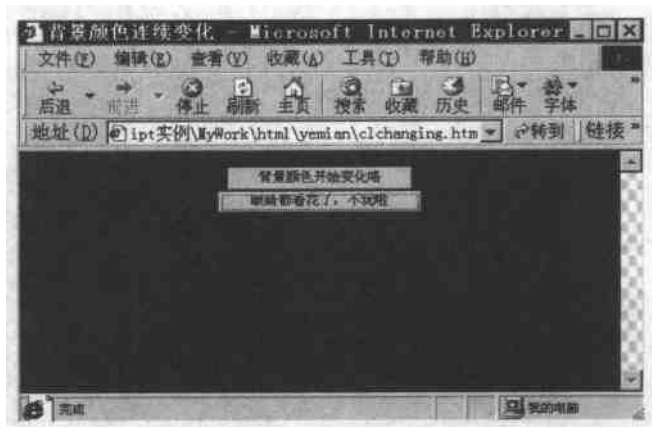


图 27.2 背景颜色保持为黑色不变

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\yemian\clchanging.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

程序先定义了两个变量 COLOR 和 woot, 用来确定背景的颜色值, 代码如下:

```
var COLOR = 999999
var woot = 0
```

随后的函数 loopBackground()通过改变 COLOR 的值持续不断地改变背景颜色。代码如下:

```
function loopBackground()
{
    if (COLOR > 0)
    {
        document.bgColor = '#' + COLOR
        COLOR -= 111111
        loopID = setTimeout("loopBackground()",10)
    }
    else
    {
        document.bgColor = '#000000'
        woot += 10
        COLOR = 999999
        COLOR -= woot
        loopID = setTimeout("loopBackground()",10)
    }
}
```

它让 COLOR 值每次减少 111111。由 999999 减少到 0 或小于 0 以后, 再将 COLOR 赋值为 999999, 并将其减去 woot 后作为新的背景颜色, 再重复上述过程。而 woot 的值也在变化, 所以 COLOR 值持续减少 111111 时背景颜色的初值也是不同的。

函数 stoploop()的作用是使背景颜色变成黑色, 然后停止颜色的变化, 代码如下:

```
function stoploop()
{
    document.bgColor = '#000000';
    clearTimeout(loopID);
}
```

4. 将光盘内 sample\yemian\clchanging.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

在此部分，程序创建了两个按钮，并设置它们 **onClick** 事件触发的函数分别设为 **loopBackground()**和 **stoploop()**。这样，当单击它们时，可使背景颜色开始变化或停止变化，代码如下：

```
<center>
<form NAME="background">
<input type="button" VALUE="背景颜色开始变化咯" onClick="loopBackground()">
<br>
<input type="button" value="眼睛都看花了，不玩啦" onClick="stoploop()">
</form>
</center>
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例28 时间决定背景颜色

【效果说明】

本例中，页面特效程序的背景颜色由时间决定。在浏览器中运行程序，如图 28.1 所示。



图 28.1 背景颜色为白色

当用户单击“获取新的背景颜色时”按钮，页面背景颜色就会发生变化。如图 28.2 所示。



图 28.2 背景颜色为绿色

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\yemian\timecolor.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

在 JavaScript 脚本程序中，函数 randombackground() 利用另一个函数 getColor() 来设置页面背景颜色，代码如下：

```
function randombackground()  
{  
    document.bgColor = getColor()  
}
```

函数 getColor() 先得到当前时间的秒数，代码如下：

```
currentdate = new Date()  
backgroundcolor = currentdate.getSeconds()
```

然后利用这个值来确定颜色值，代码如下：

```
if (backgroundcolor > 44)  
{ backgroundcolor = backgroundcolor - 45  
}  
else if (backgroundcolor > 29)  
{ backgroundcolor = backgroundcolor - 30  
}  
else if (backgroundcolor > 15)  
{ backgroundcolor = backgroundcolor - 16  
}
```

backgroundcolor 在 0~15 之间时，程序给它赋予了特定的颜色值，代码如下：

```
if (backgroundcolor == 0 )  
{ return "olive";  
}  
else if (backgroundcolor == 1 )  
{ return "teal";  
}  
...  
else if (backgroundcolor == 14 )  
{ return "white";  
}
```

```
else if (backgroundColor == 15 )  
{ return "silver";  
}
```

4. 将光盘内 sample\yemian\timecolor.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

程序在这里创建了一个按钮，并把它的 onClick 事件设置为触发函数 random background()。这样，单击这个按钮后，程序就会调用 randombackground()函数，从而使背景颜色发生改变，代码如下：

```
<center>  
<form>  
<input type="button" value="获取新的背景颜色" onClick="randombackground()">  
</form>  
</center>
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例29 背景颜色表

【效果说明】

在本例中，可以自由地改变面板背景颜色。在浏览器中运行此程序，结果如图 29.1 所示，当用户单击窗口最上面一排按钮时，背景颜色就会变成按钮上所标识的颜色。

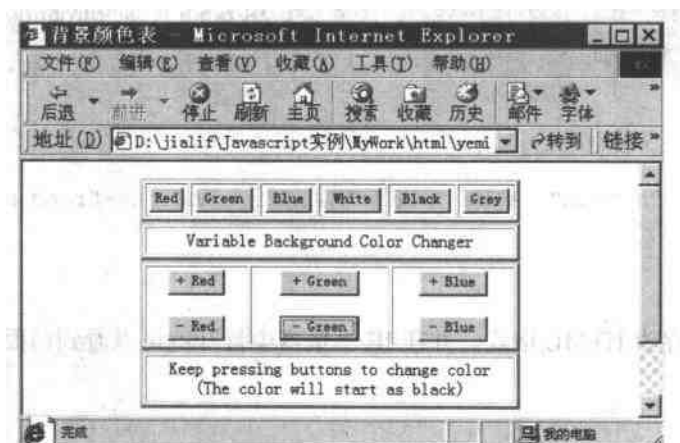


图 29.1 改变背景颜色界面

当用户单击下面两排按钮时，背景颜色就会增加或减少相应的色素，如图 29.2 所示。

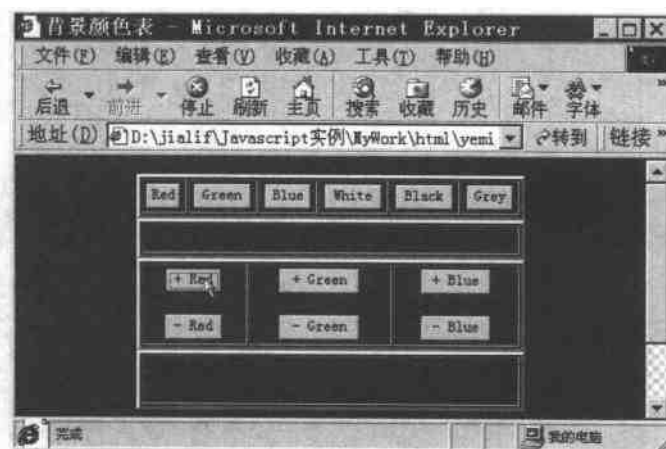


图 29.2 连续点击“+ Red”按钮

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\yernian\colortable.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

函数 `changeBackground(hexNumber)` 的作用是把页面背景颜色指定为参数 `heNumber` 的值，代码如下：

```
function changeBackground(hexNumber)
{
    document.bgColor=hexNumber
}
```

函数 `num2hex(num)` 的作用是把参数 `num` 转换成 16 进制数，代码如下：

```
function num2hex(num)
{
    if (num==15) return "f";
    else if (num==14) return "e";
    else if (num==13) return "d";
    else if (num==12) return "c";
    else if (num==11) return "b";
    else if (num==10) return "a";
    else if (num==9) return "9";
    else if (num==8) return "8";
    else if (num==7) return "7";
    else if (num==6) return "6";
    else if (num==5) return "5";
    else if (num==4) return "4";
    else if (num==3) return "3";
    else if (num==2) return "2";
    else if (num==1) return "1";
    else return "0";
}
```

下面的变量用来帮助确定背景颜色，具体作用将在函数 `changeBackground2()` 中讲解。

```
prefix="#" //颜色值前的标识
rnum1=0 //颜色的第 1 位数值：红色的第 1 位
rnum2=0 //颜色的第 2 位数值：红色的第 2 位
gnum1=0 //颜色的第 3 位数值：绿色的第 1 位
```

```

gnum2=0      //颜色的第4位数值: 绿色的第2位
bnum1=0      //颜色的第5位数值: 蓝色的第1位
bnum2=0      //颜色的第6位数值: 蓝色的第2位
hexNumber2="#000000"; //改变背景颜色前, 背景颜色的初值
rcount=0;    //用来确定红色的第1位数值
gcount=0;    //用来确定绿色的第1位数值
bcount=0;    //用来确定蓝色的第1位数值

```

函数 `changeBackground2(number)` 使背景颜色的某一色素(红、绿或蓝)逐渐改变, 采用的方法是增大或减小该色素的第1位数值。

参数 `number` 用来确定改变哪种色素的值, 它的6种取值对应于改变色素的6个按钮。

以增加红色色素为例(此时 `number` 的值为1)。用 `rcount` 来确定 `rnum1` 的值, 然后增加 `rcount` 的值, 下次运行时 `rnum1` 的值就会增加。代码如下:

```

if(number == 1)
{
    rnum1=rcount%16;
    if (rcount < 15)
    {
        rcount=rcount+1;
    }
}

```

减小红色色素的方法与此相反, 就是减小 `rcount` 的值, 代码如下:

```

if(number == 4)
{
    rnum1=rcount%16;
    if (rcount > 0)
    {
        rcount=rcount-1;
    }
}

```

改变其他色素的方法与此类似, 这里就不再一一解释。

最后, 将得到的颜色各位的值组合在一起作为背景的颜色值, 代码如下:

```

hexNumber2 =
    prefix+num2hex(rnum1)+num2hex(rnum2)+num2hex(gnum1)
    +num2hex(gnum2)+num2hex(bnum1)+num2hex(bnum2);
document.bgColor=hexNumber2

```

4. 将光盘内 sample\yemian\colortable.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

在这里程序创建了改变背景颜色的界面，主要部分就是改变颜色的按钮。把这些按钮的 onClick 事件驱动指向相应改变颜色的函数。当单击这些按钮时，颜色就相应地改变。

```
<center>
<form method="POST" name="background">
<table width=350 border="3" cellpadding="3">
<tr>
<td Align=center><input type="button" value="Red"
onclick="changeBackground('#FF0000')"></td>
...
<td Align=center><input type="button" VALUE="Grey"
onclick="changeBackground('#C0C0C0')"></td>
</tr></table>
<table width=350 border="3" cellpadding="3">
<tr><td><center>Variable Background Color Changer</center></td>
</tr></table>
<table width=350 border="3" cellpadding="3">
<tr><td Align=center>
<input type="button" value="+ Red" onclick ="changeBackground2(1)"><P>
<input type="button" value="- Red" onclick ="changeBackground2(4)">
</td>
...
</tr>
</table>
<table width=350 border="3" cellpadding="3">
<tr><td><center>Keep pressing buttons to change color<br>
(The color will start as black)</center></td>
</tr></table></form></center>
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

第5篇 文本特效

案例30 降落的文本

【效果说明】

本例是一个简单而精巧的文本特效程序。在浏览器中运行这个程序，页面上有两行文本从上往下慢慢降落，如图 30.1 所示。



图 30.1 文本降落

当下降到浏览器底部时，文本停止移动，如图 30.2 所示。



图 30.2 文本静止不动

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\dropdown\text.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

在此部分, 程序先进行了浏览器版本的判断, 代码如下:

```
browser = (((navigator.appName == "Netscape")
    && (parseInt(navigator.appVersion) >= 3 ))
    || ((navigator.appName == "Microsoft Internet Explorer")
    && (parseInt(navigator.appVersion) >= 4 )))
```

当浏览器是 Netscape 3.0 以上版本时, 此句的前一半为“真”; 当浏览器是 Internet Explorer 4.0 以上版本时, 后一半为“真”。由于此句的前后两部分是做“或”(||)运算, 因此, 只要当这二者之一为真时, 变量 browser 的值就为“真”。

```
ie4 = ((navigator.appName == "Microsoft Internet Explorer")
    && (parseInt(navigator.appVersion) >= 4 ))
ns4 = ((navigator.appName == "Netscape")
    && (parseInt(navigator.appVersion) >= 4 ))
```

上面第一句用来判断浏览器是否为 Internet Explorer 4.0 以上版本; 第二句用来判断浏览器是否为 Netscape 3.0 以上版本。

由于 Internet Explorer 与 Netscape 在表示对象的方式上有所不同, 程序为了能使两种表达方式统一起来, 特定义变量 layerRef 和 styleRef, 代码如下:

```
if (ns4)
{
    layerRef="document.layers";
    styleRef="";
}
else
{
    layerRef="document.all";
    styleRef=".style";
}
```

函数 `moveText (which, start, finish, step, speed)` 用于移动文本对象。参数 `which` 表示移动的文本对象的代号, `start` 和 `finish` 分别表示文本对象的起始位置和终止位置; `step` 表示每次移动的像素值, `speed` 表示两次移动的时间间隔。

```
function moveText(which, start, finish, step, speed)
{
    if (ie4 || ns4)
    {
        if (start < finish)
        {
            eval(layerRef + '[' + which + ']' + styleRef + '.top = start');
            start=start+step;
            setTimeout("moveText('"+which+"','"+start+"','"+finish+"',
                               "+step+", "+speed+")", speed);
        }
    }
}
```

技巧: 如果要使文本对象从 `start` 位置移动到 `finish` 位置后再从头开始移动, 只需在第二个 `if` 语句后加上下面一段程序:

```
else{
    start=0;
    setTimeout("moveText('"+which+"','"+start+"','"+finish+"',
                               "+step+", "+speed+")", speed);
}
```

4. 将光盘内 `sample\text\dropdown.txt` 中的“Part2”部分拷贝到 `<body>` 与 `</body>` 标记之间。

此段代码中程序定义了一个文本对象 `text`, 它是一个文本链接“Welcome to 北京——2008”, 代码如下:

```
<div id=text style="position: absolute; top: -100; left: 45%; z-index: 0">
<table border=0><tr><td ALIGN=center>
<font face="Helvetica" size=-1 color="#00ff00">
<a><b>Welcome To</b>
</a><br>
<a href="..\blank\blank.htm"
onmouseover="window.status='北京——2008';return true;"
onmouseout="window.status='';return true;"
>
```

```
<b>北京——2008</b></a></font></td></tr></table>  
</div>
```

提示：读者可以在这里改变链接对象和链接文本。

这里设置了文本链接的 `onMouseover` 和 `onMouseout` 事件驱动，作用是当鼠标指针移到链接上时，在状态栏中显示“北京——2008”字样，当鼠标指针移开时，上述字样消失。

5. 将 `<body>` 标记换成如下语句：

```
<body onload="moveText('text',-80,220,2,2)">
```

当加载此文档时，程序就调用 `moveText()` 函数，移动文本对象 `text`。

提示：读者可以在此改变文本对象的初始位置和终止位置，以及移动的速度。例如：`moveText('text',50,300,5,10)`。

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

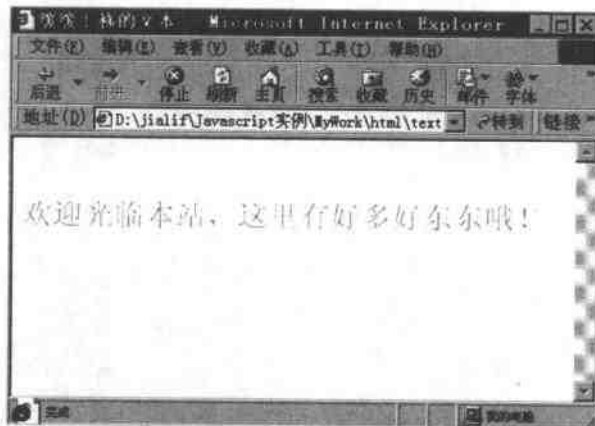
案例31 缓缓上移的文本

【效果说明】

本例是一个常用的文本特效程序。在浏览器中运行此程序，页面上有一行文本从下往上逐渐移动。如图 31.1 所示。



(1)



(2)

图 31.1 文本上升

当文本上升到浏览器顶部时，文本重新开始移动。

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\textscroller.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

在这里，程序创建了一个文本对象 `bgtick`，并给它赋予样式 `ticker`。这一样式是在 `<head>...</head>` 部分定义的。

```
<div id="bgtick" style="position:align:center" class="ticker">
欢迎光临本站，这里有好多好东东哦！</div>
```

4. 将光盘内 sample\text\textscroller.txt 中的“Part2”部分拷贝到<head>与</head>标记之间。

程序在此部分首先定义了应用于文本对象 `bgtick` 的样式 `ticker`。代码如下：

```
.ticker
{
    position: absolute;
    font-family:Verdana;
    color:AAAAAA;
    font-size:30px;
    letter-spacing:1;
}
```

在 JavaScript 脚本部分，程序定义了变量 `y`。变量 `y` 将被用来确定文本对象的位置。

```
var y=400
```

函数 `initObjects()` 用于初始化文本对象的位置，然后调用 `scroll()` 函数，代码如下：

```
function initObjects()
{
    if (document.all)
    {
        bgticker = document.all.bgtick.style
        bgticker.posLeft = 10
        bgticker.posTop = 400
    }
    if (document.layers)
    {
        bgticker = document.bgtick
        bgticker.left = 50
        bgticker.top = 300
    }
}
```

```
    }  
    scroll()  
}
```

函数 scroll() 实现移动文本的功能, 代码如下:

```
function scroll()  
{  
    if (y > -100)  
    {  
        y--  
        if (document.all) {bgticker.posTop = y;}  
        if (document.layers) {bgticker.top = y;}  
    }  
    else  
    {  
        y=400  
    }  
    var timer = setTimeout('scroll()',20)  
}
```

前面已经定义了变量 y 的值为 400, 所以开始时 y>-100 为真, y 的值减 1。然后 y 的值作为新的文本对象的位置赋给它; 当 y 值减小到-100 时, 将 y 重新赋为 400。从而使程序重新开始运行。

5. 将<body>标记换成如下语句:

```
<body onLoad="initObjects()">
```

这样, 当加载此文档时, 程序就调用 initObjects() 函数, 使程序开始运行。

6. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例32 飘动的文本

【效果说明】

本例是一个有趣的文本特效程序。在浏览器中运行该程序，页面的左上角出现一个文本框。文本框内的文本时而列队前进，时而像一面旗帜一样飘来飘去。如图 32.1 和图 32.2 所示。

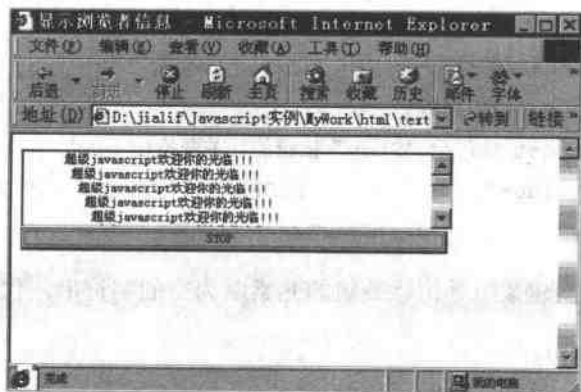


图 32.1 文本列队前进

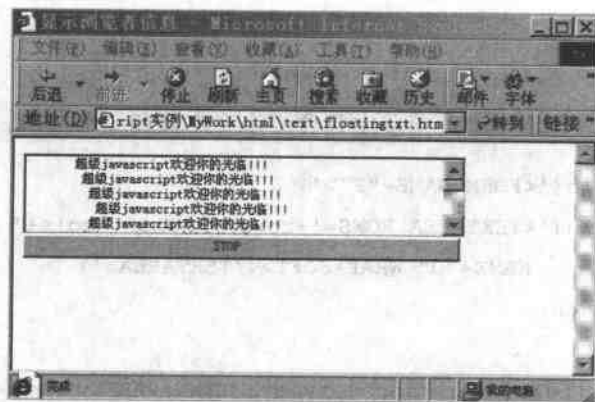


图 32.2 文本飘动

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\floatingtxt.txt 中的代码全部拷贝到<body>与</body>标记之间。

这个文本特效程序的设计思想是：定义一些字符串数组，并按照一定的规律给它们赋值，然后显示在文本框内，这样就组成了一定的图案。再把这些字符串作一些适当的改变，重新显示出来后，这些图案就动起来了。

脚本程序首先定义了如下一些变量：

```
var ltext="超级 javascript 欢迎你的光临!!!" /*显示信息*/
var spc=10 /*间距*/
var speed=50 /*速度*/
var times=2 /*显示次数*/
/* 显示的范围 */
var rows=5 /*文本区域的行数*/
var cols=60 /*文本区域的列数*/
var addstopbutton=true /* "Stop" 按钮的开关变量*/
var stopbuttonvalue="          STOP          " /* "Stop" 按钮*/
var startvalue="          START          " /* "Start" 按钮 */
```

函数 array() 用于创建数组并把该数组的元素赋为空的字符串，代码如下：

```
function array(n)
{
    this.length = n
    for (i=1; i <= n; i++){this[i] ="";}
    return this
}
```

然后程序创建了一个表单，此表单中包含一个文本域和一个按钮。

```
document.writeln('<FORM NAME="F">')
document.writeln('<TEXTAREA ROWS='+rows+' COLS='+cols+'
                    NAME="D" WRAP=SOFT></TEXTAREA>')
if (addstopbutton)
{
    document.writeln('<br><input type=BUTTON VALUE="'+stopbuttonvalue+'
                      ONCLICK="stopstart()" NAME=SS>')
}
document.writeln('</form>')
```

然后程序开始构造字符串数组。这里，程序定义了数组 lb：

```
var l=text.length
var lb=new array(l)
```

接着，对 lb 数组进行如下的处理：

```
lb[0]=text;
lb[1]=text;
for (j=1;j<l;j++)
{
    for (i=1;i<=l;i++)
    {
        lb[j]=lb[j]+lb[j-1].charAt(i);
        if (i==l)
        {
            lb[j]=lb[j]+lb[j-1].charAt(0)
        }
    }
}
```

这样，后一个 lb 字符串比前一个都要多一个字符。显示出来就是如图 32.3 的效果：

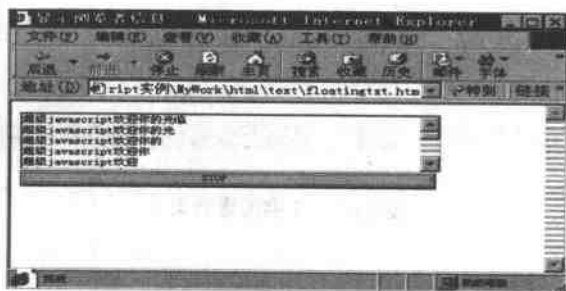


图 32.3 文本以阶梯的形状出现

然后，程序设置字符串数组 lr 的后一个比前一个多一个空格，而前一个又比后一个多一个空格，这样显示出来就是前面图 32.2 的形状。

```
lr[0]=ltext;
lr[spc*2]=ltext;
for (i=1;i<=spc;i++)
{
    lr[i]=" "+lr[i-1]
}
for (i=spc*2-1;i>=spc+1;i--)
{
    lr[i]=" "+lr[i+1]
}
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例33 文字逐个闪耀

【效果说明】

本例中的文本特效程序可以用来发布需要引起浏览者注意的内容。在浏览器中运行此程序，页面上方出现一行文字，而且这些文字逐个地闪耀，就好像霓虹灯广告牌一样。如图 33.1 所示。



图 33.1 文字闪耀效果

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\sparkonebyone.txt 中的代码全部拷贝到<body>与</body>标记之间。

程序首先定义了如下一些全局变量：

```
text = "欢迎光临网页制作特效站"; //显示的文字
color1 = "blue"; //文字的颜色
color2 = "red"; //转换的颜色
fontsize = "6"; //字体大小
speed = 100; //转换速度 (1000 = 1 秒)
```

然后程序建立了一个层或容器，其 id 为 a，代码如下：

```
if (navigator.appName == "Netscape")
```

```

{
    //建立层
    document.write("<layer id=a visibility=show></layer><br><br><br>");
}
else
{
    //建立容器
    document.write("<div id=a></div>");
}

```

函数 `changeCharColor()` 按浏览器是 Netscape 或 Internet Explorer 的不同进行了不同的操作, 但其基本思想是一样的: 将某字符串 `text` 中的某一个字符如 `text[i]` 用红色输出, 其余的都蓝色输出, 然后让 `i` 的值加 1, 重新执行函数 `changeCharColor()`, 这样下一个字符就变成了红色。当 `i` 值达到最大 (等于 `text` 的长度), 就给 `i` 值赋 0。无限重复这个过程就达到了文字依次闪耀的效果。

程序中定义了一个变量 `i`, 来记录改变颜色的字符编号。

```
i=0;
```

函数先对 Netscape 浏览器的情况做了处理。

用前面定义的变量来设置将要显示的文字的字体和颜色, 代码如下:

```

document.a.document.write("<center><font face=arial size ="
                           + fontsize + "><font color=" + color1 + ">");

```

注意这里的 `color1` 是蓝色 `blue`。

然后函数把要显示的字符一个一个添在后面, 如果是字符 `text[i]`, 就把它变成红色, 代码如下:

```

for (var j = 0; j < text.length; j++)
{
    if(j == i)
    {
        //将 text[i] 变成红色
        document.a.document.write("<font face=arial color=" + color2 + ">"
                                   + Text.charAt(i) + "</font>");
    }
    else
    {
        document.a.document.write(text.charAt(j));
    }
}

```


将 HTML 语句补充完整，并关闭文档输出流，代码如下：

```
document.a.document.write('</font></font></center>');  
document.a.document.close();
```

如果是 Internet Explorer 浏览器，函数先将生成页面的 HTML 语言放在字符串 str 中，然后把它显示出来，代码如下：

```
if (navigator.appName == "Microsoft Internet Explorer")  
{  
    str = "<center><font face=arial size=" + fontsize + "><font color=" + color1 + ">";  
    for (var j = 0; j < text.length; j++)  
    {  
        if( j == i)  
        {  
            str += "<font face=arial color=" + color2 + ">" + text.charAt(i) + "</font>";  
        }  
        else  
        {  
            str += text.charAt(j);  
        }  
    }  
    str += "</font></font></center>";  
    a.innerHTML = str;  
}
```

接着，改变 i 的值，代码如下：

```
if(i == text.length)  
{ i=0 ;  
}  
else  
{ i++;  
}
```

这样，函数 changeCharColor() 就算结束了。

最后，设置每隔 100 毫秒执行一次 changeCharColor() 函数。

```
setInterval("changeCharColor()", speed);
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例34 旋转变换的文本

【效果说明】

本例是一个非常有趣的文本特效程序。在浏览器中运行该程序，浏览器中有一串文本不停地旋转着，字体和颜色也不停地变化。如图 34.1 和图 34.2 所示。



图 34.1 文本旋转 (1)



图 34.2 文本旋转 (2)

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\text\xztxt.txt 中的代码全部拷贝到<head>与</head>标记之间。

此程序的设计思路是将要显示的字符串中的每个字符分别放在不同容器中，然后改变容器的位置同时改变字体大小和文字颜色，来实现文字的特效。

程序先创建了如下几个变量：

```
Phrase="欢迎你的光临" //旋转的字符串
Taille=40;           //字体的基数值
Midx=100;            //位置的基数值
Nb=Phrase.length;    //旋转字符串的长度
```

然后程序用<div>...</div>标识为每个字符创建了一个容器，并给它们分配了不同的 id 值，代码如下：

```
Balises=""
for (x=0;x<Nb;x++)
{
    Balises=Balises + '<div Id=L' + x + ' STYLE="width:3;
                        font-family: Courier New;font-weight:bold;
                        position:absolute;top:40;left:50;z-index:0">' +
                        Phrase.charAt(x) + '</div>'
}
document.write (Balises);
```

在 Alors()函数之前，程序定义了如下 3 个变量，来初始化文本容器的位置和颜色：

```
Alpha=5;
I_Alpha=0.05;
Decal=0.5;
```

函数 Alors()为每一个文本容器设置了相应的位置和颜色，代码如下：

```
for (x=0;x<Nb;x++)
{
    Alpha1=Alpha+Decal*x;
    Cosine=Math.cos(Alpha1);
    Ob=document.all("L"+x);
    Ob.style.posLeft=Midx+100*Math.sin(Alpha1)+50;
    Ob.style.zIndex=20*Cosine;
    Ob.style.fontSize=Taille+25*Cosine;
    Ob.style.color="rgb("+ (27+Cosine*80+50) + ", "+
                    (127+Cosine*80+50) + ",0)";
}
```

然后改变 Alpha 的值，这样，下次运行时每个文本的位置和颜色也会有所改变：

```
Alpha=Alpha-I_Alpha;
```

注意：这里采用的正弦和余弦函数是周期函数，所以程序运行的效果也是周期性变化的。

设置每隔 10 毫秒运行一次 Alors() 函数，代码如下：

```
Time=window.setInterval("Alors()",10);
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例35 文字效果变幻

【效果说明】

本例是一个简单的文本特效程序。在浏览器中运行此程序，页面的左上角有一行文字不停地变换着效果。如图 35.1 和图 35.2 所示。



图 35.1 文字效果变幻 (1)



图 35.2 文字效果变幻 (2)

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\text\xgbh.txt 的“Part1”部分拷贝到<head>与</head>标记之间。

程序首先定义了一个名为 glowtext 的 CSS 样式，代码如下：

```
<style>
<!--
#glowtext{
filter:glow(color=red,strength=2);
width:100%;
}
-->
</style>
```

函数 glowit2()用来改变文本对象 glowtext 的效果——filters 滤镜的值，代码如下：

```
function glowit2()
{
if (document.all.glowtext.filters[0].strength==2)
    document.all.glowtext.filters[0].strength=1
else
    document.all.glowtext.filters[0].strength=2
}
```

其中 glowtext 为<body>...</body>区域定义的文本对象。

与函数 glowit2()一样，函数 glowit(which)也是用来改变文本对象的效果，但是它有一个参数 which，此参数用来标识文本对象 glowtext 的序号，代码如下：

```
function glowit(which)
{
if (document.all.glowtext[which].filters[0].strength==2)
    document.all.glowtext[which].filters[0].strength=1
else
    document.all.glowtext[which].filters[0].strength=2
}
```

提示：如果程序用同一个 id 值来建立多个文本区域，当要引用其中某一个对象时就需用 idname[num]的形式了。

函数 startglowing()对文本对象 glowtext 的个数进行了判断，并根据个数的不同调用 glowit()函数或 glowit2()函数，代码如下：

```
function startglowing()
{
```

```
if (document.all.glowtext&&glowtext.length)
{
    for (i=0;i<glowtext.length;i++)
        eval('setInterval("glowit('+i+')",150)')
}
else if (glowtext)
{
    setInterval("glowit2()",150)
}
}
```

最后，程序把函数 `startglowing()` 设定为 `onload` 事件的驱动。这样，当加载此文档时，就会运行 `startglowing()` 函数。

```
if (document.all)
    window.onload=startglowing
```

4. 将光盘内 `sample\text\xgbh.txt` 中的“Part2”部分拷贝到 `<body>` 与 `</body>` 标记之间。

此部分程序创建了一个文本对象 `glowtext`：

```
<span id="glowtext">文字效果变幻</span>
```

提示：由于程序只定义了一个 `glowtext` 对象，所以函数 `startglowing()` 将不执行 `if` 部分的代码。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例36 字符消隐特效

【效果说明】

本例是一个字符消隐特效程序。在浏览器中运行该程序，页面中间的一串文字先逐渐变浅，然后逐渐加深，如图 36.1 所示。

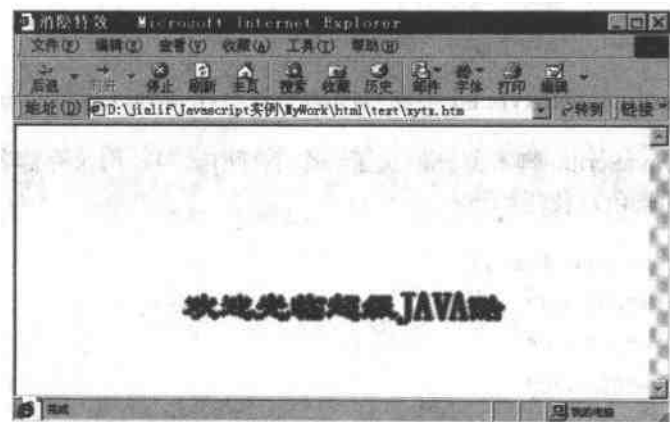


图 36.1 文字效果变浅

接着出现另外一行文字，重复上述过程。如图 36.2 所示。

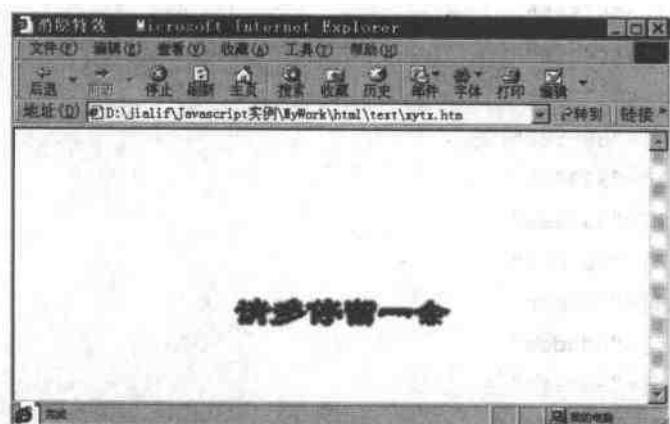


图 36.2 另一行文字消隐

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\xytx.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

程序在此部分创建了一个容器, id 为 glowdiv, 此容器就是程序<head>...</head>部分要制作消隐特效的文本容器。

```
<div id="glowdiv" style="position:absolute;visibility:visible;  
    width:600px;text-align:center; top:185px;left:70px;font-family:求书;  
    font-size:30pt;color:000000">  
</div>
```

4. 将光盘内 sample\text\xytx.txt 中的“Part2”部分拷贝到<head>与</head>标记之间。

程序首先在 JavaScript 脚本部分创建了一个字符串数组, 用来存储各种颜色, 此数组将用来改变文字的颜色, 代码如下:

```
var textcolor= new Array()  
textcolor[0]="000000"  
textcolor[1]="000000"  
textcolor[2]="000000"  
textcolor[3]="111111"  
textcolor[4]="222222"  
textcolor[5]="333333"  
textcolor[6]="444444"  
textcolor[7]="555555"  
textcolor[8]="666666"  
textcolor[9]="777777"  
textcolor[10]="888888"  
textcolor[11]="999999"  
textcolor[12]="aaaaaa"  
textcolor[13]="bbbbbb"  
textcolor[14]="cccccc"  
textcolor[15]="dddddd"  
textcolor[16]="eeeeee"  
textcolor[17]="ffffff"  
textcolor[18]="ffffff"
```

接着, 程序定义了一个数组来存储将要显示的文本, 代码如下:

```
var message = new Array()  
message[0]="欢迎光临超级 JAVA 酷"  
message[1]="请多停留一会"
```

```
message[2]="你会有更多的收获"  
message[3]="请再次光临"
```

将文本的序号和颜色的序号设置为 0，代码如下：

```
var i_strength=0  
var i_message=0
```

函数 `glowtext()` 用来给文字附加发光效果，代码如下：

```
function glowtext()  
{  
    if(document.all)  
    {  
        if (i_strength <=17)  
        {  
            glowdiv.innerText=message[i_message]  
            document.all.glowdiv.style.filter="glow(color=  
                "+textcolor[i_strength]+"", strength=4)"  
            i_strength++  
            timer=setTimeout("glowtext()",100)  
        }  
        else  
        {  
            clearTimeout(timer)  
            setTimeout("degloowtext()",1500)  
        }  
    }  
}
```

在上述代码中，先判断 `i_strength` 的值是否已达到颜色数组 `textcolor` 的上限 17，若没有达到，则对文字 `message[i_message]` 施加颜色为 `textcolor[i_strength]` 的发光效果，然后减小 `i_strength` 的值，并再次执行 `glowtext()` 函数；若已达到 `textcolor` 的上限，则调用 `degloowtext()` 函数。

`degloowtext()` 函数的作用与 `glowtext()` 相反，它是将发光效果的参数——发光颜色 `textcolor` 的值由 `textcolor[17]` 逐渐变为 `textcolor[0]`，其代码如下：

```
function degloowtext()  
{  
    if(document.all)  
    {  
        if (i_strength >=0)
```

```
{
    glowdiv.innerText=message[i_message]
    document.all.glowdiv.style.filter="glow(color=
        "+textcolor[i_strength]+" , strength=4)"
    i_strength--
    timer=setTimeout("deglowtext()",100)
}
else
{
    clearTimeout(timer)
    i_message++
    if (i_message>=message.length)
    { i_message=0
    }
    i_strength=0
    intermezzo()
}
}
```

当 `i_strength` 的值达到颜色数组 `textcolor` 的下限 0 后, 改变文本的内容, 并调用 `intermezzo()` 函数使屏幕保持空白 1 秒钟, 代码如下:

```
function intermezzo()
{
    glowdiv.innerText=""
    setTimeout("glowtext()",1000)
}
```

5. 把 `<body>` 语句换成如下代码, 使文档被加载时开始运行文本特效程序。

```
<body bgcolor="#fef4d9" onLoad="glowtext()">
```

6. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例37 文本自动输出特效

【效果说明】

本例是一个文本自动输出的特效程序。在浏览器中运行该程序，页面的左上角有一个黑色的文本区域，字符一个一个地出现在这个文本区域内，就好像人工输入一样。如图 37.1 所示。

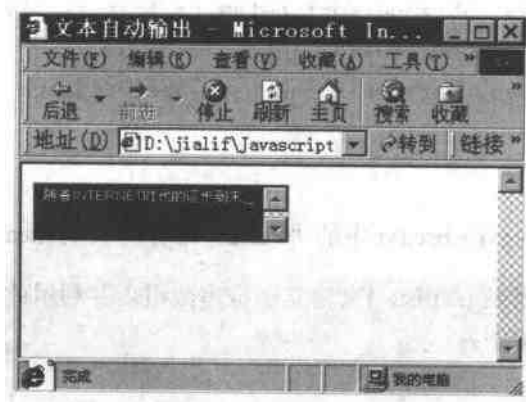


图 37.1 字符一个一个地出现

写满一行后，出现另外一行文字，然后重复上述过程。如图 37.2 所示。

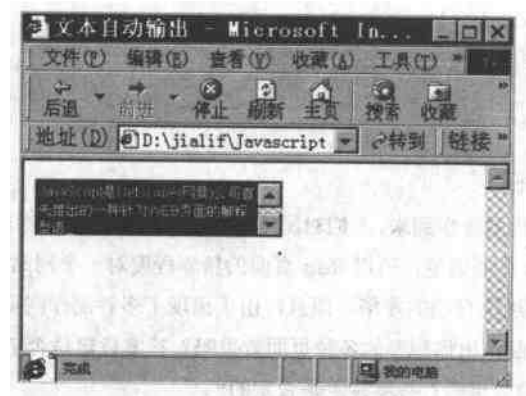


图 37.2 另一行文字出现

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\text\wbsc.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

程序在这里创建了一个文本区域 tickfield, 最初显示的文本是 The news will appear here when the page has finished loading.。但由于此页下载的速度很快, 实际上, 大家很难看见这条消息。

```
<form name=tickform>
<textarea name=tickfield rows=3 cols=38
        style="background-color:rgb(0,0,0);
color: rgb(255,255,255); cursor: default; font-family: Arial;
font-size: 12px" wrap=virtual>
The news will appear here when the page has finished loading.
</textarea>
</form>
```

4. 将光盘内 sample\text\wbsc.txt 中的“Part2”部分拷贝到<head>与</head>标记之间。

程序先创建了一个函数 textlist()来建立字符串数组, 并利用此函数把要显示的字符串存储在数组 tl 中, 代码如下:

```
var max=0;
function textlist()
{
    max=textlist.arguments.length;
    for (i=0; i<max; i++)
        this[i]=textlist.arguments[i];
}

tl = new textlist(
    " 随着 Internet 时代的逐步到来, 人们对网络的认识与感知越来越深刻",
    "上网主要是进行 Web 页面浏览, 所以 Web 页面的精彩程度对一个网站的生",
    "人通过制作个人主页展现自己的才华, 而且, 由于出现了多种制作网页的软件",
    "但是, 光用软件就成制作出你想要的各种页面效果吗? 答案肯定是否定的但是, 光用软件就成制作",
    "出你想要的各种页面效果吗? 答案肯定是否定的",
    "JavaScript 是 Netscape (网景) 公司首先推出的一种针对 web 页面的解释型语言"
);
```

然后程序定义了 3 个变量: x 用来标识显示的文本; pos 用来表示把文本的哪一部分显示出来; l 表示第一个字符串 tl[0]的长度。

```
var x = 0; pos = 0;
var l = tl[0].length;
```

函数 `textticker()` 用于将字符串显示在文本区域中，代码如下：

```
function textticker()
{
    document.tickform.tickfield.value = t1[x].substring(0, pos) + "_";
    if(pos++ == 1)
    {
        pos = 0;
        setTimeout("textticker()", 2000);
        if(++x == max)
            x = 0;
        l = t1[x].length;
    }
    else
        setTimeout("textticker()", 50);
}
```

每运行一次，将 `pos` 加 1，这样下次显示的字符就多了一个。当一个字符串显示完后（`pos` 等于 `max`），改变 `x` 的值，换一个字符串显示。

5. 把 `<body>` 标记换成如下代码，使文档被加载时开始运行文本特效程序。

```
<body bgcolor="#fef4d9" OnLoad="textticker()">
```

技巧：可以随意更改这里的背景颜色。例如将背景改成蓝色，只需将上面的 `bgcolor="#fef4d9"` 改写成 `bgcolor="#0000ff"` 即可。

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例38 文本颜色渐变

【效果说明】

本例是一个文本颜色渐变的程序。在浏览器中打开后，页面中出现一串文本，并且这串文本的颜色先由红色逐渐变为黄色，再逐渐变成紫色。如图 38.1 所示。



图 38.1 文本颜色渐变

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\ysjb.txt 中的代码全部拷贝到<body>与</body>标记之间。

程序首先用函数 MakeArray(n)创建了一个数组，并将数组元素按顺序赋值，数组中元素的个数由 n 来确定，代码如下：

```
function MakeArray(n)
{
    this.length=n;
    for(var i=1; i<=n; i++) this[i]=i-1;
    return this
}
```

然后利用此函数创建了一个包含 16 个元素的数组 hex[]，并将第 11~16 个元素分别赋值为 A、B、C、D、E 和 F。这个数组将被用来进行十六进制的转化。

下面的函数 ToHex(x)用来把十进制的数字转化为十六进制，在后面将用来生成显示字

字符串的颜色。

先用“整除”获得十六进制数中的高位，代码如下：

```
var high=x/16;
```

接着把它转化成字符，并取前两位，代码如下：

```
var s=high+"";
s=s.substring(0,2);
```

再用十六进制表示，存作 left，代码如下：

```
high=parseInt(s,10);
var left=hex[high+1];
```

然后计算十六进制的低位，代码如下：

```
var low=x-high*16;
```

用同样的方法将低位转换成字符串，代码如下：

```
s=low+"";
s=s.substring(0,2);
low=parseInt(s,10);
var right=hex[low+1];
```

最后输出结果，代码如下：

```
var string=left+""+right;
return string;
```

函数 rainbow(text)用于输出颜色渐变的文本。

先设定颜色变化的范围，代码如下：

```
color_d1=255;
```

接着，根据所要显示的字符串的长度来计算相邻字符间的颜色变化，代码如下：

```
mul=color_d1/text.length;
```

然后函数开始逐个地显示字符，代码如下：

```
for(i=0;i < text.length;i++)
{
    color_d1=255*Math.sin(i/(text.length/3));
    color_h1=ToHex(color_d1);
    color_d2=mul*i;
```



```
color_h2=ToHex(color_d2);  
document.write("<FONT COLOR='#FF"+color_h1+color_h2+"'>"  
+text.substring(i,i+1)+'</font>');  
}
```

技巧：这里固定了前两位颜色为 FF，中间两位的变化规律是正弦变化，后两位则按顺序递增。读者在这里可以自己设置颜色变化的规律。例如将中间两位改成按余弦周期性变化。只需把语句 `color_d1=255*Math.sin(i/(text.length/3))` 改为 `color_d1=255*Math.cos(2*Math.PI/(text.length))`。

接下来，程序就调用 `rainbow()` 函数来显示颜色渐变的文本。

```
<script>  
<!--  
{rainbow("--> HELLO! LET'S ROCK !!! YOUR MESSAGE GOES HERE.  
THE LONG THE BETTER!<!--");}  
/-->  
</script>
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例 39 文本弹跳特效

【效果说明】

本例是一个文本在页面内弹跳的特效程序。在浏览器中运行此程序，页面中有一行文本不停地上下弹跳。如图 39.1 和图 39.2 所示。

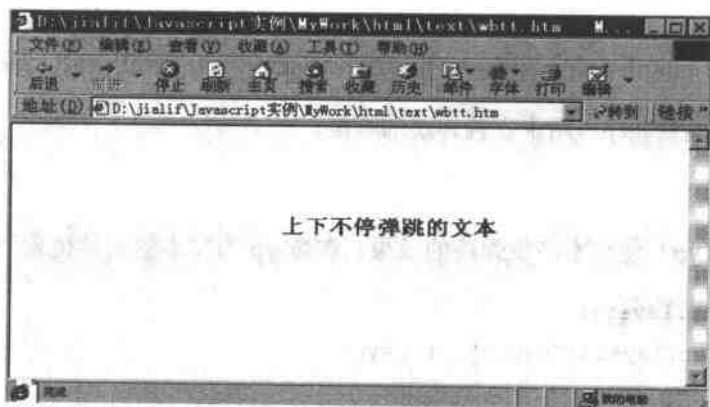


图 39.1 文本上升

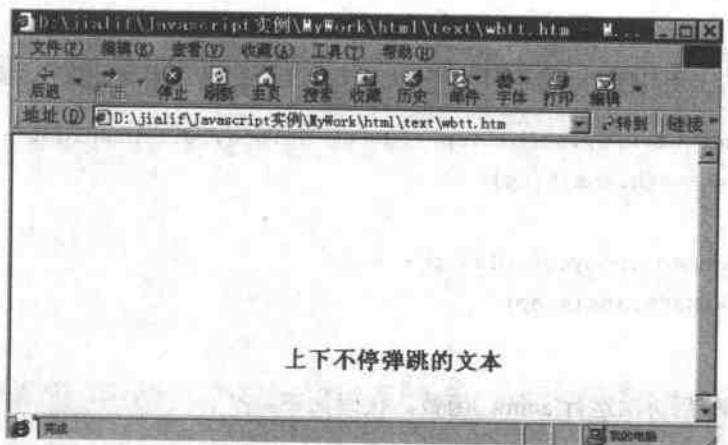


图 39.2 文本下降

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。

3. 将光盘内 sample\text\wbtt.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

程序在此部分创建了一个文本对象 naps，并设置了它的样式，代码如下：

```
<div id="naps" style="position: absolute; top: 159px; width: 400px;
        height: 78px; left: 215px"><font size="5">
<b>上下不停弹跳的文本</b>
</font>
</div>
```

4. 将光盘内 sample\text\wbtt.txt 中的“Part2”部分拷贝到<head>与</head>标记之间。

首先定义了变量 step，用于设置弹跳的幅度：

```
step = 4;
```

函数 anim (yp) 使文本产生弹跳的效果，参数 yp 为文本的初始位置。

```
if(document.layers)
{ document.layers["naps"].top=yp;
}
else
{ document.all["naps"].style.top=yp;
}
```

当文本距离上下边界 60 像素时，将 step 取反，使文本往相反的方向运动，代码如下：

```
if(yp>document.body.scrollTop + document.body.clientHeight - 60)
{ step = -Math.abs(step)
}
if(yp<document.body.scrollTop + 60)
{ step = Math.abs(step)
}
```

间隔 10 毫秒后再次运行 anim() 函数，代码如下：

```
setTimeout('anim('+(yp+step)+')', 10);
```

函数 start() 用于设置文本横向的位置，并调用 anim() 函数，代码如下：

```
done = 0;
function start()
{
if(done) return
done = 1;
```

```
if(navigator.appName=="Netscape")
{
    document.napis.left=innerWidth/2 - 145;
    anim(60)
}
else {
    napis.style.left=280;
    anim(document.body.scrollTop + 60)
}
```

5. 把<body>语句换成如下代码，使文档被加载时开始运行文本特效程序。

```
<body onload="start()">
```

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例40 元素周期表

【效果说明】

本例是一个非常有用的文本特效程序。如图 40.1 所示,从这个元素周期表中可以看到元素的种类、顺序、状态、原子量、电子分布、熔点和沸点等信息,还能知道哪些元素是人造的,哪些是天然存在的。



图 40.1 元素周期表

要知道具体元素的信息,只需单击该元素。如图 40.2 所示为 Fe 元素的信息。



图 40.2 Fe 元素的信息

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\text\periodic.txt 中的“Part 1”部分拷贝到<head>与</head>标记之间。

程序用 HTML 语言构建元素周期表的框架，用文字链接来表示表中的各元素。以第一个元素 H 为例，代码如下：

```
<td bgcolor="#FFFF00">  
<a href="javascript: fillitin('Hydrogen','1','1.0079','1',  
                                '1s1','-259.14 C',' -252.87 C');">  
<fontsize="2">H  
</font></a><font size="2"> </font></td>
```

单击该链接时，调用 fillitin() 函数将该元素的信息填入相应的文本框中，代码如下：

```
function fillitin(Name, AtomicNumber, AtomicWeight, Shells, FillingOrbital,  
MeltingPoint, BoilingPoint)  
{  
    show(Name);  
    document.PeriodicTable.Name.value=Name;  
    document.PeriodicTable.AtomicNumber.value=AtomicNumber;  
    document.PeriodicTable.AtomicWeight.value=AtomicWeight;  
    document.PeriodicTable.Shells.value=Shells;  
    document.PeriodicTable.FillingOrbital.value=FillingOrbital;  
    document.PeriodicTable.MeltingPoint.value=MeltingPoint;  
    document.PeriodicTable.BoilingPoint.value=BoilingPoint;  
    window.setTimeout("prepare()", 3500);  
}
```

其中函数 show(Name) 的作用是将元素的名字显示在状态栏中，代码如下：

```
function show(txt)  
{  
    window.status = txt;  
}
```

函数 prepare() 的作用是在状态栏显示“欢迎”字样，代码如下：

```
function prepare()  
{  
    window.setTimeout("window.status='欢迎使用本程序'",500);  
}
```

单击元素表中的某链接后，先执行函数 `show(Name)`，状态栏中显示元素的名字，间隔 3500 毫秒后执行 `prepare()` 函数，再过 500 毫秒后显示“欢迎”字样。

函数 `fillitin()` 中的 `PeriodicTable` 是元素周期表的名字，`Name`、`AtomicNumber`、`AtomicWeight` 等符号是 `PeriodicTable` 下方文本输入框的名字。

4. 将 `<body>` 标记换成如下语句：

```
<body bgcolor="#fef4d9" onload="prepare();">
```

这样，当文档被加载时，程序就执行 `prepare()` 函数，在状态栏中显示“欢迎”字样。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

第6篇 页面导航

案例41 动态导航

【效果说明】

本例是一个简单而实用的导航程序。在浏览器中运行此程序，导航链接会一个一个在页面中显示出来，并逐渐向上移动。如图 41.1 所示。



图 41.1 导航链接逐条出现

当导航链接全部出现后，接着会向上移动并逐渐消失，直至所有链接全部消失。如图 41.2 所示。

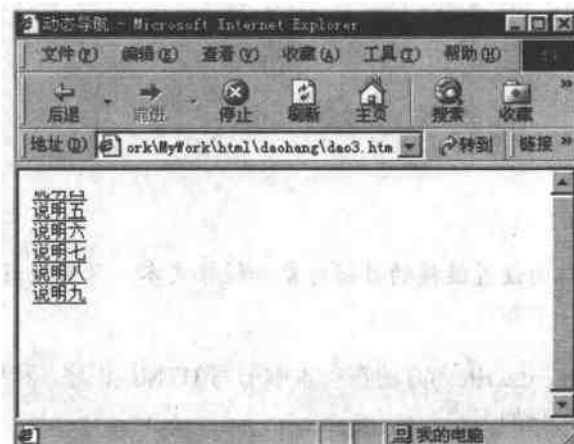


图 41.2 导航链接逐条消失

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\dtdh.txt 中的代码全部拷贝到<body>与</body>标记之间。

在 JavaScript 脚本部分，程序首先定义两个数组 link 和 text。其中，数组 link 用于存储链接的目标对象，而数组 text 用于存储链接的说明文本：

```
var index = 9
link = new Array(8);
text = new Array(8);

link[0] = 'blank.htm'
link[1] = 'blank.htm'
link[2] = 'blank.htm'
link[3] = 'blank.htm'
link[4] = 'blank.htm'
link[5] = 'blank.htm'
link[6] = 'blank.htm'
link[7] = 'blank.htm'
link[8] = 'blank.htm'

text[0] = '说明一'
text[1] = '说明二'
text[2] = '说明三'
text[3] = '说明四'
text[4] = '说明五'
text[5] = '说明六'
text[6] = '说明七'
text[7] = '说明八'
text[8] = '说明九'
```

提示：读者可以自由设置链接的目标对象和链接文本，只需将 link 数组和 text 数组元素的值更改一下即可。

然后程序用 document.write() 方法在脚本中书写 HTML 程序，利用<marquee>标记创建一串可移动的文本，代码如下：

```
document.write("<marquee scrollamount='1' scrolldelay='100'")
```

```
direction= 'up' width='150' height='150'>");
```

scrollDelay 属性设置的是两次滚动操作的间隔时间,这里设置的是 100 毫秒,即 0.1 秒;scrollAmount 设置的是每次滚动操作的位移,这里设置的是 1 个像素,即滚动一次后文本移动 1 个像素;direction 用来设置滚动的方向,可取的值为: up、down、left 和 right; width 和 height 分别设置滚动字幕的宽度和高度,当字幕滚动超出这个范围时,字幕就会逐渐消失(滚出界外)。

提示:读者可以在这里选择合适的滚动方向。只需将 direction 属性的值改为 up、down、left 和 right 四者之一即可。

接下来,程序将链接的目标对象和链接文本用 document.write() 方法写入页面,代码如下:

```
for (i=0;i<index;i++)
{
    document.write ("<a href="+link[i]+" target='_blank'>");
    document.write (text[i] + "</a><br>");
}
```

这样就在<marquee>标记内部建立了 9 个链接。
最后添上<marquee>标记的另一半,代码如下:

```
document.write ("</marquee>")
```

4. 将文件保存为 HTML 格式,并在 IE 浏览器中打开就可以看到前面所述的效果。

案例42 隐 现 导 航

【效果说明】

本例是一个制作精巧的导航程序。在浏览器中运行此程序，浏览器的左上角会出现一白色的亮条。亮条很快由白转黑，此时会发现有一个超链接隐藏在其中。如图 42.1 所示。



图 42.1 亮条转暗链接出现

过两、三秒钟后，上述动作重复，不过出现的却是另一个链接。如图 42.2 所示。



图 42.2 出现另一个链接

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\yxdh.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

程序先用 CSS 语法定义了两个 id: tickertape 和 subtickertape, 两个 class (类): subtickertapefont 和 subtickertapefonta, 代码如下:

```
<style>
<!--
#tickertape{
position:relative;
layer-background-color:black;
width:400;
height:12;
; font-family: "宋体"; font-size: 9pt)

#subtickertape{
background-color:black;
position:absolute;
border: 1px solid black;
width:400;
height:12;
; font-family: "宋体"; font-size: 9pt
}

.subtickertapefont{
font:bold 9pt "宋体";
text-decoration:none;
color:white;
}

.subtickertapefont a{
color:white;
text-decoration:none;
; font-family: "宋体"; font-size: 9pt}
-->
</style>
```

4. 将光盘内 sample\daohang\yxdh.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

程序首先利用<div>...</div>创建了两个嵌套的图层，并把它们用 tickertape 和 subtickertape 表示，代码如下：

```
<div id="tickertape">
<div id="subtickertape" class="subtickertapefont">
初始化...
</div>
```

```
</div>
```

注意：由于程序运行很快，这里的字符串“初始化...”在屏幕上一闪而过，因而很难看见。

在 JavaScript 脚本部分，程序首先定义了链接在屏幕上出现的时间，代码如下：

```
var speed=3000
```

提示：读者在这里可以随意改变链接停留的时间，只需将 speed 赋予一个新值即可。

然后创建数组 news 来存放超链接的内容，代码如下：

```
var news=new Array()  
news[0]="<a href='index1.htm'>连接 1</a>"  
news[1]="<a href='index2.htm'>连接 2</a>"  
news[2]="<a href='index3.htm'>连接 3</a>"  
news[3]="<a href='index4.htm'>连接 4</a>"  
news[4]="<a href='index5.htm'>连接 5</a>"  
news[5]="<a href='index6.htm'>连接 6</a>"  
news[6]="<a href='index7.htm'>连接 7</a>"
```

提示：读者可以自由设置链接的目标对象和链接文本。

接着把前面定义的图层赋给 tickerobject 对象。后面将对这个对象的背景颜色作连续快速的变化，以达到前面所述的背景特效。

为了达到变化背景的效果，程序创建了函数 BgFade()。它有 7 个参数，前 6 个参数是颜色值，分别代表“红”、“绿”、“蓝”，“红”、“绿”、“蓝”。前 3 个颜色表示变化前的颜色值，后面 3 个表示变化后的颜色值。第 7 个参数 steps 代表颜色变化的步数，即每个链接出现的过程中，背景颜色变化的次数。函数 BgFade() 将这些参数赋给了另外 6 个变量，这里只是为函数 RunFader() 作准备，代码如下：

```
function BgFade(red1, grn1, blu1, red2, grn2, blu2, steps)  
{  
    sred = red1;  
    sgrn = grn1;  
    sblu = blu1;  
    ered = red2;  
    egrn = grn2;  
    eblu = blu2;  
    inc = steps;  
    step = 0;  
    RunFader();  
}
```

```
}
```

函数 `RunFader()` 首先定义了两个变量 `epct` 和 `spct`, 作为调整颜色的系数, 代码如下:

```
var epct = step/inc;  
var spct = 1 - epct;
```

函数根据这两个系数获得背景颜色的数值。这里函数对 Netscape 浏览器和 IE 浏览器分别做了处理, 代码如下:

```
if (document.layers)  
{  
    tickerobject.bgColor =  
    Math.floor(sred * spct + ered * epct)*256*256 +  
    Math.floor(sgrn * spct + egrn * epct)*256 +  
    Math.floor(sblu * spct + eblu * epct);  
}  
else  
{  
    tickerobject.backgroundColor=  
    Math.floor(sred * spct + ered * epct)*256*256 +  
    Math.floor(sgrn * spct + egrn * epct)*256 +  
    Math.floor(sblu * spct + eblu * epct);  
}
```

增加 `step` 的值, 改变系数, 重新运行 `RunFader()` 函数, 以改变背景颜色, 代码如下:

```
if ( step < inc )  
{  
    setTimeout('RunFader()',50);  
}  
step++;
```

提示: 这里可以调整颜色变化的快慢。只需将 `setTimeout()` 中的数值 50 换成别的数值即可。

函数 `update()` 首先调用 `BgFade()` 来改变背景颜色, 代码如下:

```
BgFade(0xff,0xff,0xff, 0x00,0x00,0x00,10);
```

提示: 读者可以在这里调整颜色值和颜色变化的步数。

接着函数改变链接的文本和对象, 对 Netscape 浏览器和 IE 浏览器分别操作, 代码如下:

```
if (document.layers)
```

```
{
    document.tickertape.document.subtickertape.document.write(
        '<span class="subtickertapefont">'+news[i]+'</span>')
    document.tickertape.document.subtickertape.document.close()
}
else
{
    document.all.subtickertape.innerHTML=news[1]
}
```

改变*i*的序号，重新运行 `update()` 函数，代码如下：

```
if (i<news.length-1)
{ i++
}
else
{ i=0
}
setTimeout("update()",speed)
```

5. 将<body>标记换成如下内容：

```
<body bgcolor="#fef4d9" onload="if (document.all||document.layers)
update()">
```

这样，当加载网页时，`onload` 事件就会被触发，从而调用 `update()` 函数。

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例43 下拉式导航菜单

【效果说明】

本例是一个精美而实用的导航程序。在浏览器中运行此程序，页面中就会出现一个名为“点这里看本站栏目”的链接。如图 43.1 所示。



图 43.1 导航链接

用鼠标单击这个链接，一个下拉导航菜单便由模糊到清楚逐渐显示出来。如图 43.2 所示。

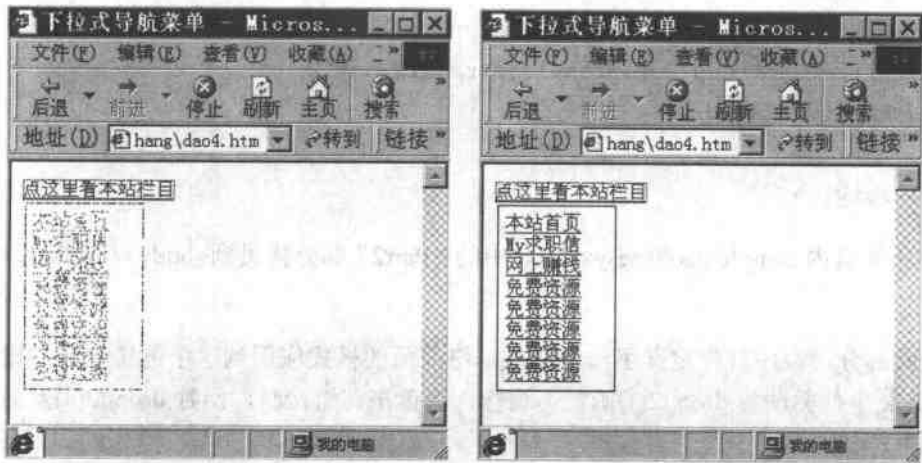


图 43.2 下拉导航菜单逐条出现

当鼠标在链接上再次单击，或在菜单之外的区域单击时，下拉导航菜单便消失。

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\yxdh.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

在这里，程序定义了3个CSS样式ID：iewrap、iewrap2 和 dropmenu03，代码如下：

```
<style>
<!--
#iewrap
{
position:relative;height:30px
}

#iewrap2
{
position:absolute
}

#dropmenu03
{
filter:revealTrans(Duration=1.5,Transition=12)
visibility:hide
}
--></style>
```

4. 将光盘内 sample\daohang\yxdh.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

在此部分，程序首先定义了一个链接，将前面的样式应用到这个链接上，并设置链接的 onClick 事件驱动为 dropit2()函数。这样，当单击此链接时，函数 dropit2()就会运行，代码如下：

```
<ilayer id="dropmenu01" height=35px>
<layer id="dropmenu02" visibility=show>
<span id="iewrap">
<span id="iewrap2" onClick="dropit2();event.cancelBubble=true;
return false">
```

```

<font face="宋体"><a href="#">点这里看本站栏目</a></font></span>
</span>
</layer>
</ilayer>

```

在 JavaScript 脚本程序的起始处定义了开关变量 `enableeffect`，并把它赋为 `true`，代码如下：

```
var enableeffect=true
```

接着创建一个数组 `selection`，用于存储导航菜单的菜单项。读者在这里可以设置自己的菜单链接和菜单内容。代码如下：

```

var selection=new Array()
selection[0]='<font face="宋体"><a href="time1.htm">
本站首页</a></font><br>'
selection[1]='<font face="宋体"><a href="time2.htm">
My 求职信</a></font><br>'
selection[2]='<font face="宋体"><a href="time3.htm">
网上赚钱</a></font><br>'
selection[3]='<font face="宋体"><a href="ye1.htm">免费资源</a></font><br>'
selection[4]='<font face="宋体"><a href="ye2.htm">免费资源</a></font><br>'
selection[5]='<font face="宋体"><a href="ye3.htm">免费资源</a></font><br>'
selection[6]='<font face="宋体"><a href="ye4.htm">免费资源</a></font><br>'
selection[7]='<font face="宋体"><a href="zh5.htm">免费资源</a></font>'

```

然后程序利用 `document.write()` 方法把这 8 个菜单项按照样式 `dropmenu03` 排列起来。注意这里设置 `visibility` 属性为 `hidden`，所以这些菜单项开始是看不见的。程序将通过改变这个样式以达到菜单项由模糊到清晰的效果。代码如下：

```

<div id="dropmenu03" style="position:absolute;left:10;top:0;
    layer-background-color:seashell;background-color:seashell;
    width:100;visibility:hidden;border:1px solid black;padding:0px">
<script language="JavaScript1.2">
<!--
if (document.all)
{
    dropmenu03.style.padding='6px'
}
for (i=0;i<selection.length;i++)
{
    document.write(selection[i])

```

```
}  
//-->  
</script>  
</div>
```

函数 **dropit2()** 首先设置了菜单项的位置，代码如下：

```
dropmenu03.style.left=  
    document.body.scrollLeft+event.clientX-event.offsetX  
dropmenu03.style.top=  
    document.body.scrollTop+event.clientY-event.offsetY+18
```

如果此时菜单项是不可见的，就改变它的样式，使它可见，代码如下：

```
if (dropmenu03.style.visibility=="hidden")  
{  
    if (enableeffect)  
        dropmenu03.filters.revealTrans.apply()  
    dropmenu03.style.visibility="visible"  
    if (enableeffect)  
        dropmenu03.filters.revealTrans.play()  
}
```

如果此时菜单项是可见的，则调用 **hidemenu()** 函数，使它消失，代码如下：

```
else  
{  
    hidemenu()  
}
```

hidemenu() 函数的作用是使样式 **dropmenu03** 失效，并使菜单项消失，代码如下：

```
function hidemenu()  
{  
    if (enableeffect)  
        dropmenu03.filters.revealTrans.stop()  
    dropmenu03.style.visibility="hidden"  
}
```

说明：奇数次点击时，菜单项的 **visibility** 属性是 **hidden**，在执行语句 **dropmenu03.style.visibility="visible"** 时属性变为 **visible**；偶数次点击时，菜单项的 **visibility** 属性是 **visible**，在执行 **hidemenu()** 函数时属性变为 **hidden**。

当鼠标在菜单项之外的区域单击时，菜单项也会消失，此功能由下面一行程序实现：

```
document.body.onclick=hidemenu
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例44 层叠式导航菜单

【效果说明】

本例是一个常用的菜单导航程序。在浏览器中运行此程序，页面中就会出现两个链接，如图 44.1 所示。



图 44.1 导航链接

单击其中的任意一个链接，相应的导航菜单就会显示出来。如图 44.2 所示。

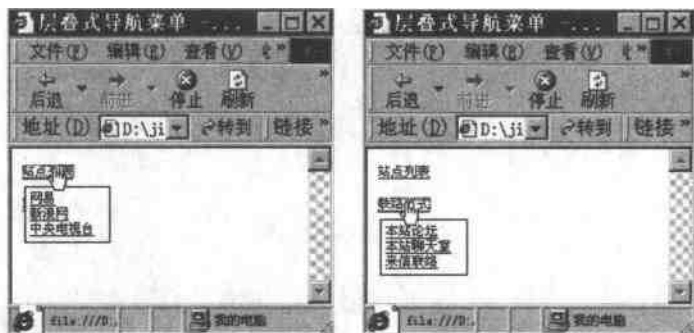


图 44.2 下拉导航菜单逐条出现

当鼠标在链接上再次单击，或在菜单之外的区域单击时，菜单项即消失。

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\cddh.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

在 JavaScript 脚本部分程序首先创建了数组 menu1 和 menu2 来存储导航菜单的菜单项。读者在这里可以设置自己的菜单链接和菜单内容。代码如下：

```
//菜单 1 的内容
var menu1=new Array()
menu1[0]='<a href=http://www.163.net>网易</a><br>'
menu1[1]='<a href=http://www.sina.com.cn>新浪网</a><br>'
menu1[2]='<a href=http://www.cctv.com>中央电视台</a><br>'
//菜单 2 的内容
var menu2=new Array()
menu2[0]='<a href=http://vs_server.njvat.net/bbs>本站论坛</a><br>'
menu2[1]='<a href=http://vs_server.njcatv.net/aspchitchat>
    本站聊天室</a><br>'
menu2[2]='<a href=mailto:baron@njcatv.net>来信联络</a><br>'
```

然后，程序定义了几个 CSS 样式 ID: iewrap1、iewrap2、dropmenu0 和 dropmenu0。

4. 将光盘内 sample\daohang\cddh.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

在此部分，程序定义了两个链接，将前面的样式应用到该链接上，并设置链接的 onClick 事件驱动为 dropit2() 函数。这样，当该链接被单击时，函数 dropit2() 就会运行，代码如下：

```
<!--菜单 1 开始-->
<ilayer height=35px>
<layer visibility=show>
<span class=iewrap1>
<span class=iewrap2 onClick="dropit2(dropmenu0);
    event.cancelBubble=true;return false">
<font face=宋体>
<a href="#" onClick="if(document.layers)
    return dropit(event, 'document.dropmenu0')">
站点列表</a></font>
</span></span>
</iayer></ilayer><br>
<!--菜单 1 结束-->
<!--菜单 2 开始-->
<ilayer height=35px>
<layer visibility=show>
<span class=iewrap1>
<span class=iewrap2 onClick="dropit2(dropmenu1);
```

```

        event.cancelBubble=true;return false">
<font face=宋体>
<a href="#" onClick="if(document.layers)
        return dropit(event, 'document.dropmenu1')">
联络方式</a></font>
</span></span>
</layer></ilayer><br>
<!-------菜单2结束----->

```

接着程序利用 document.write() 方法把两个菜单的菜单项按照一定的样式排列起来。由于这里设置 visibility 属性为 hidden，所以这些菜单项开始是看不见的。程序将通过改变样式使菜单项显示出来。代码如下：

```

<div id=dropmenu0 style="position:absolute;left:0;top:0;
        layer-background-color:#CDECFF5;background-color:#CDECFF5;
        width:80;visibility:hidden;border:1px solid black;padding:0px">
<script language="JavaScript1.2">
if (document.all)
dropmenu0.style.padding="4px"
for (i=0;i<menu1.length;i++)
document.write(menu1[i])
</script>
</div>
<div id=dropmenu1 style="position:absolute;left:0;top:0;
        layer-background-color:#CDECFF5;background-color:#CDECFF5;
        width:80;visibility:hidden;border:1px solid black;padding:0px">
<script language="JavaScript1.2">
if (document.all)
dropmenu1.style.padding="4px"
for (i=0;i<menu2.length;i++)
document.write(menu2[i])
</script>
</div>

```

再回头介绍一下函数 dropit2()，它首先设置了菜单项的位置，代码如下：

```

themenu.style.left=document.body.scrollLeft+event.clientX-event.offsetX
themenu.style.top=document.body.scrollTop+event.clientY-
        event.offsetY+18

```

如果此时菜单项是不可见的，就改变它的样式，使它可见，并使它的层叠顺序加 1，代码如下：

```
if (themenue.style.visibility=="hidden")
{
    themenu.style.visibility="visible"
    themenu.style.zIndex=zindex++
}
```

如果此时菜单项是可见的，则调用 `hidemenu()` 函数，使它消失，代码如下：

```
else
{
    hidemenu()
}
```

`hidemenu()` 函数的作用是使菜单项消失，代码如下：

```
function hidemenu(whichone)
{
    if (window.themenue)
        themenu.style.visibility="hidden"
}
```

当鼠标在菜单项之外的区域单击时，菜单项也会消失，此功能由下面的一行程序实现：

```
document.body.onclick=hidemenu
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例45 目录式导航菜单

【效果说明】

本例是一个目录式菜单导航程序。在浏览器中运行此程序，页面中就会出现两个链接，如图 45.1 所示。

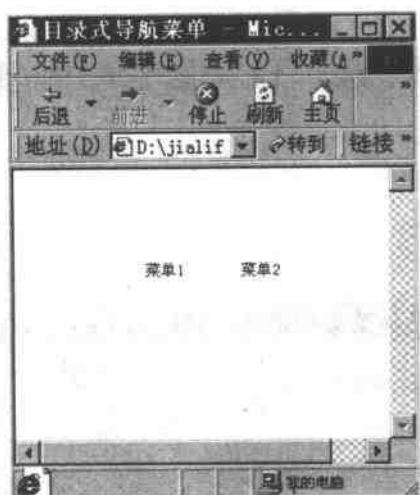


图 45.1 初始页面

鼠标指针移动到任意一个链接上时，相应的下拉菜单就会显示出来，如图 45.2 所示。



图 45.2 出现下拉菜单

当鼠标指针移动到链接以外的区域时，下拉菜单自动消失。

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\mldh.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

此部分的脚本程序创建了两个函数 out() 和 out1()。这两个函数的作用是当鼠标指针移动到菜单项以外的区域时，使菜单项消失。代码如下：

```
function out()
{
    if(window.event.toElement.id!="menu"&&
        window.event.toElement.id!="link")
        menu.style.visibility="hidden";
}
function out1()
{
    if(window.event.toElement.id!="menu1"&&
        window.event.toElement.id!="link")
        menu1.style.visibility="hidden";
}
```

程序代码中的 menu、link 和 menu1 是后面将要定义的菜单和菜单项的代号。out() 和 out1() 函数先判断此时鼠标指针是否在这些菜单或菜单项上，若不在，就把它们的 visibility 属性设置为不可见“hidden”。

4. 将光盘内 sample\daohang\mldh.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

程序用<div>...</div>定义了一个图层，id 设置为 back。设置它的 onmouseover 事件驱动函数为 out()，这样，当鼠标指针移动到菜单项之外的区域时，out() 函数就会运行，从而使相应的菜单项不可见。

程序接着用...定义了一个文本项“菜单 1”，并将它的 onmouseover 事件驱动设置为“menu.style.visibility='visible’”。这样，当鼠标指针移动到文本上时，菜单项的 visibility 属性就会设置为 visible，从而使菜单项可见。

程序紧接着定义了菜单项 menu。注意这里设置 visibility 属性为 hidden，所以这些菜单项开始是看不见的。代码如下：

```
<div id="back" onmouseout="out()" style="position:absolute;top:80;
```

```
        left:110;width:160;height:40;z-index:1;visibility:visible;">
<span id="menubar" onmouseover="menu.style.visibility='visible'">
<font color=red size=2>菜单 1
</span>
<div border=1 id="menu" style="position:absolute;top:15;
    left:0;width:60;height:10;z-index:2;visibility:hidden;">
<a id="link" href="javascript:if(confirm(
    'http://www.csdn.net/javascript/rjxz.htm \n\n 这个文件不能通过 Teleport
    Pro 取回, 因为 服务器报告找不到这个文件. \n\n
    你想从服务器打开它吗?'))
    window.location='http://www.csdn.net/javascript/rjxz.htm'
    tppabs="http://www.csdn.net/javascript/rjxz.htm">
    软件下载
</a>
.....
</div>
</div>
```

后面“菜单 2”的设计方法与“菜单 1”相同, 这里就不再赘述。

5. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例46 移动导航菜单

【效果说明】

本例是一个构思精巧的菜单导航程序。在浏览器中运行此程序，页面中就会出现一个链接，如图 46.1 所示。



图 46.1 初始页面

单击该链接，相应的导航菜单自左向右移动，直至完全显示出来，如图 46.2 所示。



图 46.2 导航菜单自左向右移动

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\yddh.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

此部分程序先用 document.writeln()方法建立了一个链接“导航菜单”，设置它的链接目标为 slide()函数。这样，当用户单击此链接时，程序就会去执行 slide()函数。代码如下：

```
<script language = "javascript">
<!--
if(ns4||ie4)
document.writeln('<div id="buttons"><a href="javascript:slide()"
onclick="window.focus()">
<font face="Helvetica" size=-1>
导航菜单
</font></a></div>')
//-->
</script>
```

然后程序建立了导航菜单的菜单项，它的 id 为 ownmenu。注意，这里设置的 left 参数为-150，所以程序刚运行时看不见这些菜单项。代码如下：

```
<div id="ownmenu" style="position:absolute;top:120;
left:-150;width:60;z-index:100;">
<table width=120 border=0 cellpadding=0 cellspacing=0><tr><td><center>
<font face="Helvetica" size=-1>
<a href="#" onmouseover="window.status='返回主页';return true;"
onmouseout="window.status='';return true;" onclick="window.focus()">
返回主页</a><br>
.....
<a href="javascript:back()" onmouseover="window.status='关闭菜单';
return true;"
onmouseout="window.status='';return true;" onclick="window.focus()">
关闭菜单</a><br>
</font></center>
</td></tr></table>
</div>
```

其中，有一个菜单项是“关闭菜单”，它被链接到 back()函数。后面将会讲到 back()函数是如何工作的。

4. 将光盘内 sample\daohang\yddh.txt 中的“Part2”部分拷贝到<head>与</head>标记之间。

在此部分, 程序先进行了浏览器版本的判断, 代码如下:

```
ns4 = (document.layers)? true:false  
ie4 = (document.all)? true:false
```

当浏览器是 Netscape 时, document.layers 为真, document.all 为假, 从而 ns4 的值为 true, ie4 的值为 false; 反之, 当浏览器是 Internet Explorer 时, document.layers 为假, document.all 为真, 从而 ns4 的值为 false, ie4 的值为 true。

下面的 init() 函数用来初始化菜单对象 ownmenu 的位置, 它将 ownmenu 对象的 left 属性作为它的横坐标。前面提到 ownmenu 对象的 left 属性的值为 -150, 因此这里 ownmenu 对象的横坐标为 -150, 从而使它处于屏幕之外。init() 函数的代码如下:

```
function init()  
{  
    if(ns4||ie4)  
    {  
        if (ns4)  
        {  
            block1 = document.ownmenu  
        }  
        if (ie4)  
        {  
            block1 = ownmenu.style  
        }  
        block1.x = parseInt(block1.left)  
    }  
}
```

下面的 slide() 函数将菜单项 ownmenu 的横坐标值增加 5, 重复执行 slide() 函数, 从而呈现菜单项自左向右移动的效果, 代码如下:

```
function slide()  
{  
    if(ns4||ie4)  
    {  
        if (block1.x < 5)  
        {  
            block1.x += 5  
            block1.left = block1.x  
            setTimeout("slide()",20)  
        }  
    }  
}
```

```
}  
}
```

提示：读者可以在这里改变菜单项移动的速度和幅度。要改变速度，可以将 `setTimeout()` 函数中的“20”换成其他的数值；要改变幅度。可以将“`block1.x += 5`”语句中的“5”改成其他的值。

`back()` 函数将菜单项 `ownmenu` 的 `left` 属性和横坐标都赋为-150，从而使菜单项消失，代码如下：

```
function back()  
{  
    if(ns4||ie4)  
    {  
        block1.x = -150  
        block1.left = block1.x  
    }  
}
```

5. 将<body>标记换成如下代码：

```
<body onload="init();">
```

这样，当网页能加载时就执行 `init()` 函数，使程序初始化，准备运行。

6. 将文件保存为 HTML 格式，并在浏览器中打开就可以看到前面所述的效果。

案例47 导航菜单说明

【效果说明】

本例中的程序可为导航菜单增加说明。在浏览器中运行此程序，页面中就会出现一个导航菜单，如图 47.1 所示。



图 47.1 导航菜单

当鼠标指针移动到菜单项上面时，在浏览器的状态栏及菜单右下方会出现相应的说明，如图 47.2 所示。

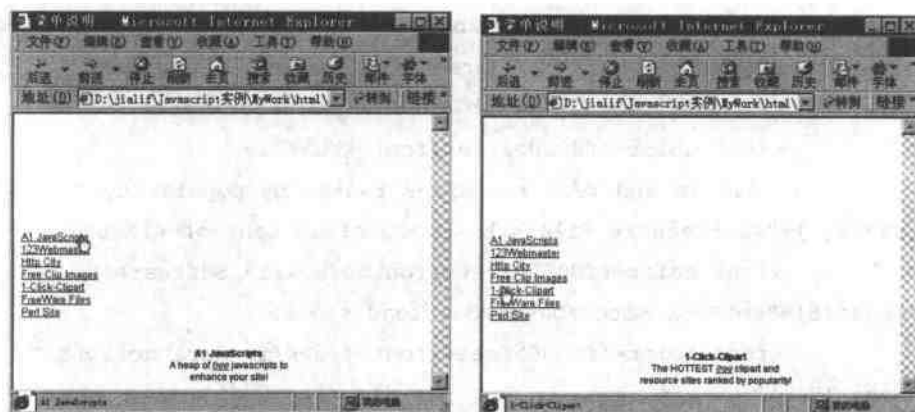


图 47.2 导航菜单有相应的说明

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\dhsm.txt 中的“Part I”部分拷贝到<head>与</head>标记之间。

程序先定义了几个字符串, 这几个字符串在后面被用来设置菜单说明的显示属性。代码如下:

```
//Set the font for the msg
msgFont='Arial,helvetica'
//Set the fontSize in px
msgFontSize="12"
//Set the fontColor
msgFontColor="black"
```

下面定义的字符串数组用来存储菜单说明文本, 并给它们附加了一些样式, 代码如下:

```
messages=new Array(7)
messages[0]="<b>A1 JavaScripts</b><br>A heap of <i><u><font color=ff0000>
    free</font></u></i> javascripts to enhance your site!"
messages[1]="<b>123Webmaster</b><br>The largest directory of <i><u>
    <font color=ff0000>free</font></u></i>
    webmaster resources on the planet!"
messages[2]="<b>HttpCITY</b><br>25 megs of <i><u><font color=ff0000>
    free</font></u></i> webspace;<br> httpcity.com/you!"
messages[3]="<b>Free Clip Images</b><br>A monstrous collection of <i><u>
    <font color=ff0000>free</font></u></i> graphics!"
messages[4]="<b>1-Click-Clipart</b><br>The HOTTEST <i><u>
    <font color=ff0000>free</font></u></i>
    clipart and resource sites ranked by popularity!"
messages[5]="<b>FreeWare Files</b><br>Download tons of <i><u>
    <font color=ff0000>free</font></u></i> Software!"
messages[6]="<b>Perl Site</b><br>Download <i><u>
    <font color=ff0000>free</font></u></i> perl scripts!"
messages[7]=""
```

接着程序为判断浏览器的类型, 定义了两个变量 ie 和 n。若浏览器是 Internet Explorer, 则 ie 为真; 否则, n 为真。代码如下:

```
ie=document.all?1:0
n=document.layers?1:0
```

函数 `changeTextInit()` 创建了一个对象区域 `oMessage`，并在此区域显示在 `<body>...</body>` 部分创建的文本对象 `divMessage`。此文本对象的内容就是在图 47.2 中页面底部显示的文字说明。函数 `ChangeTextInit()` 的代码如下：

```
function changeTextInit()
{
    if (ie || n)
        oMessage=new makeChangeTextObj('divMessage')
}
```

其中函数 `makeChangeTextObj(obj)` 用来创建一个对象，它具有 `writeref` 属性和 `writelt` 方法。`writeIt` 方法被赋予了 `b_writeIt` 函数。代码如下：

```
function makeChangeTextObj(obj)
{
    this.writeref=(n) ? eval('document.'+obj+'.document'):eval(obj);
    this.writeIt=b_writeIt;
}
```

`b_write(text)` 函数用来显示文本 `text`，代码如下：

```
function b_writeIt(text)
{
    if(n)
    {
        this.writeref.write(text)
        this.writeref.close()
    }
    if(ie)
        this.writeref.innerHTML=text
}
```

函数 `changeText()` 调用 `b_write()` 函数来显示菜单项的说明文本，其代码如下：

```
function changeText(num)
{
    if(ie || n)
        oMessage.writeIt('<span style="font-size:'+msgFontSize+'px;
        " font-family:'+msgFont+';color:'+msgFontColor+'">'
        +messages[num]+'</span>')
}
```

`onload=changeTextInit` 语句使文档被加载时就执行 `changeTextInit()` 函数。

4. 将光盘内 sample\daohang\dhsm.txt 中的“Part2”部分拷贝到<body>与</body>标记之间。

程序在此部分定义了文本对象 divMessage，代码如下：

```
<table Border=0 cellpadding=0 cellspacing=0 width="400" height="200">
<td align=center width="100%" height="100%">
<div id="divMessage" style="position:absolute; left:200; top:300">
This is the default text that will show in<br> 3.x browsers. You have to place
this with tables or <br>something instead of layers for backwards compatibility.
</td></table></div>
```

然后创建了 7 个菜单项，设置它们的 onmouseover 事件驱动为 changeText()函数。这样，当鼠标指针移动到这些菜单项上时，程序就会执行 changeText()函数，从而显示出相应的程序说明。代码如下：

```
<div id="divlinks" style="position:absolute; left:10; top:150">
<a href="#" onmouseover="changeText(0); window.status='Al JavaScripts';
return true;" onMouseout="changeText(7); window.status='';
return true;" onclick="window.focus()">
Al JavaScripts</a><br>
.....
<a href="#" onmouseover="changeText(6); window.status='Perl Site';
return true;" onMouseout="changeText(7); window.status='';
return true;" onclick="window.focus()">
Perl Site</a>
</div>
```

下面这一句代码的作用如下：当鼠标指针离开菜单项时，程序会执行 changeText(7)，而 messages[7]是空字符串，因而没有菜单说明出现。

```
onMouseout="changeText(7); window.status=''; return true;"
```

6. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例48 自动变色的链接

【效果说明】

本例是一个非常简单，但也很实用的小程序。

在浏览器中运行此程序，把鼠标指针放到页面中的链接上时，可以看到链接文字和它的背景色都变了，而当鼠标指针移出时，它们又恢复了正常。效果如图 48.1 和图 48.2 所示。

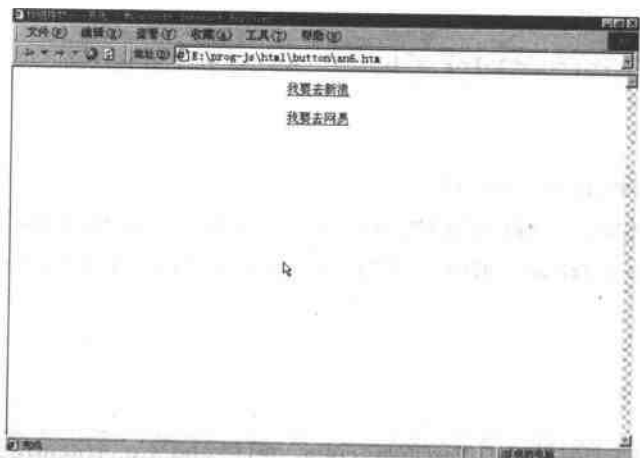


图 48.1 链接变色前

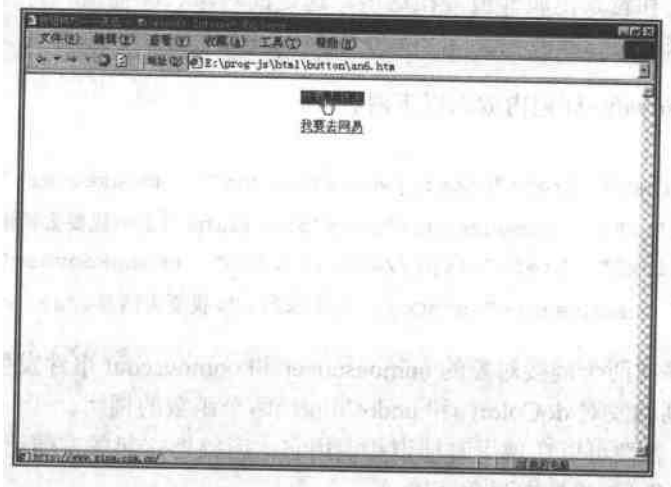


图 48.2 鼠标指针移到链接上链接变色

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。

2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。

3. 把光盘内 sample\daohang\changecolor.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">

<!-- Begin
function doColor(item, color, bg) {
item.style.color = color; // changes text color
item.style.backgroundColor = bg; // changes background
}

function undoColor(item) {
item.style.color = "#0000FF"; // sets color back to blue
item.style.backgroundColor = ""; // sets background to default
}

// End -->
</script>
```

这里可以看到两个函数 doColor(item, color, bg)和 undoColor(item), 它们分别负责改变选定的对象的颜色属性“.style.color”值和背景色属性“.style.backgroundColor”值以及恢复选定对象的颜色和背景色属性值。在这里, 选定的对象、颜色值、背景色值都是以函数参数形式传递进来的。

4. 在<body></body>标记内放入以下内容。

```
<center>
<a name="link1" href="http://www.sina.com" onmouseover="doColor(link1,
'yellow','black');" onmouseout="undoColor(link1);">我要去新浪</a><p>
<a name="link2" href="http://www.163.com" onmouseover="doColor(link2,
'blue','red');" onmouseout="undoColor(link2);">我要去网易</a> </center>
```

在这里, 程序在两个链接对象的 onmouseover 和 onmouseout 事件发生时, 即鼠标指针移入和移出时分别触发对 doColor()和 undoColor()两个函数的调用。

对象的 name 属性可以在使用时标识和调用它, 应该把它起成方便记忆和识别的名字, 如这里的 link1 和 link2 就是指两个链接。

在改变对象的颜色和背景色时, 应该使用对比强烈的两种颜色, 不然对象和背景混为一体就很难看清楚。

5. 最后, 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例49 浮动链接导航条

【效果说明】

本例是一个实用的导航程序。在浏览器中运行此程序，页面中出现 4 个文本链接，如图 49.1 所示。当鼠标移到其中任何一个链接上时，页面右下方就会出现一段文字，介绍该链接的基本内容。如图 49.2 所示。

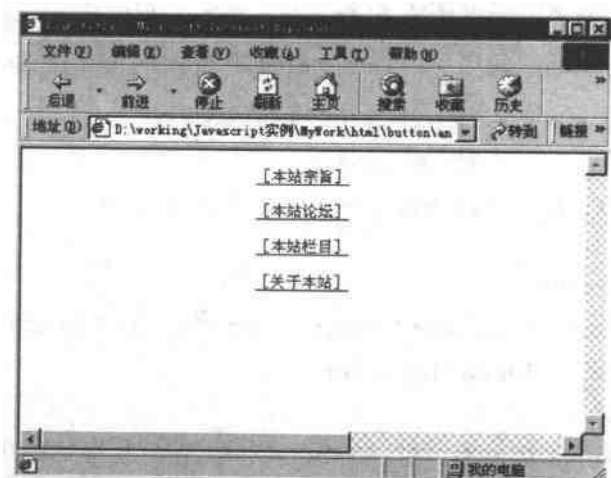


图 49.1 实例开始效果

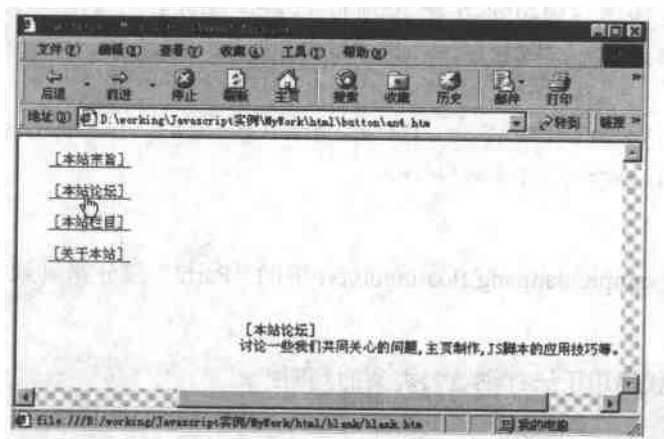


图 49.2 浮动导航说明

注意：图 49.1 和图 49.2 中的 4 个文本链接一直在页面的中间。但是在图 49.2 中为了取图的方便将文本链接移到了左边。

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\daohang\floatingdh.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

此段代码的作用是在页面上创建 4 个超链接, 当鼠标指针放在这些超链接上时, 利用 onmouseover 事件调用 don() 程序; 用 onmouseout 事件调用 doff() 程序。

```
<div id="linkex" style="position: absolute; visibility: hidden; width=80%">
< div>
<layer name="linkex" visibility="hide" width=80%>
</Layer>
<p align="center">
<a href="..\blank\blank.htm" onmouseover="don(0)" onmouseout="doff()"
target="_blank">[本站宗旨] </a><br>
<br>
<a href="..\blank\blank.htm" onmouseover="don(1)" onmouseout="doff()"
target="_blank">[本站论坛] </a><br>
<br>
<a href="..\blank\blank.htm" onmouseover="don(2)" onmouseout="doff()"
target="_blank">[本站栏目] </a><br>
<br>
<a href="..\blank\blank.htm" onmouseover="don(3)" onmouseout="doff()"
target="_blank">[关于本站] </a>
</p>
```

4. 将光盘内 sample\daohang\floatingdh.txt 中的“Part2”部分拷贝到<head>与</head>标记之间。

开头的一段代码用于进行浏览器类型的判断:

```
bname=navigator.appName;
bversion=parseInt(navigator.appVersion)
if (bname=="Netscape")
brows=true
else
brows=false
```

如果浏览器是 Netscape, 将 brows 变量设置为 true; 否则, 就将 brows 变量设置为 false。然后创建了 4 个数组: link[0]、link[1]、link[2]和 link[3], 分别来存放将要显示的浮动信息。

程序 don()先判断了浏览器是否为 Netscape 或者 Internet Explorer:

```
if ((bname=="Netscape" && bversion>=4) || (bname=="Microsoft Internet Explorer" && bversion>=4))
{
...
}
```

如果是二者之一, 程序就进一步判断是 Netscape 还是 Internet Explorer, 并且将数组 link[x]的信息赋给相应的 Layer 对象或 Document 对象显示出来, 代码如下:

```
if (brows){
with(link[x]){
document.layers['linkex'].bgColor=bgcolor;
document.layers['linkex'].document.writeln(msg);
document.layers['linkex'].document.close();
document.layers['linkex'].top=dtop;
document.layers['linkex'].left=dleft;
}
document.layers['linkex'].visibility="show";
}
else{
with(link[x]){
linkex.innerHTML=msg;
linkex.style.top=dtop;
linkex.style.left=dleft;
linkex.style.background=bgcolor;
}
linkex.style.visibility="visible";
}
```

5. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例50 跑马灯式栏目指南

【效果说明】

本例是一个非常有实用价值的 JavaScript 小程序，通过使用类似 Windows 的提示条，可以为页面增色不少。

程序运行结果如图 50.1 所示。当把鼠标指针移到页面各个栏目的链接上时，就会在旁边出现一个小小的提示条，里面用跑马灯的形式介绍各个栏目的大致内容和特点，使得浏览者先对栏目有个大致了解。如图 50.2 所示。

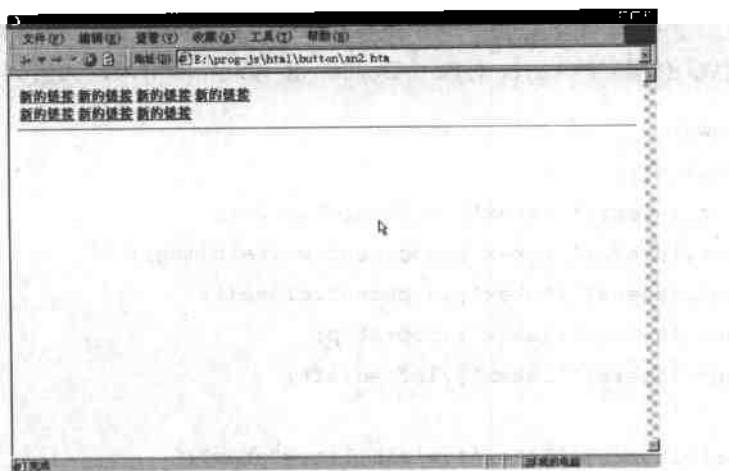


图 50.1 程序运行初始状态

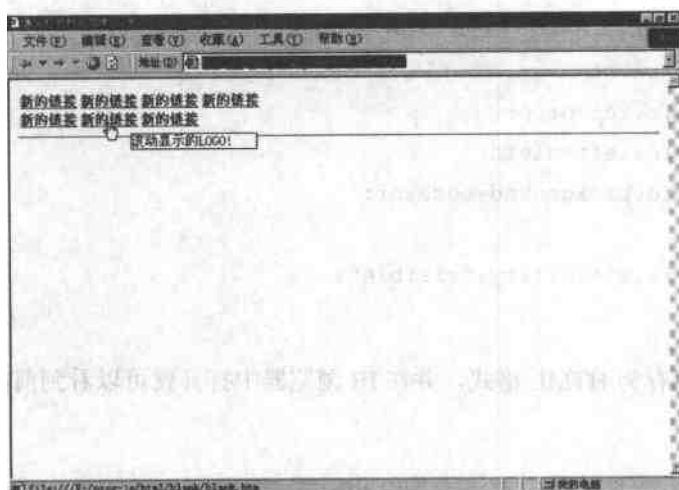


图 50.2 鼠标指针放在链接上出现跑马灯式说明

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 进入“网页”视图 (选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 把光盘内 sample\daohang\dhpmid.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

```
<script language="JavaScript">
<!--
function showtip(current,e,text)
{
if (document.all&&document.readyState=="complete")
{
document.all.tooltip.innerHTML = '<marqueestyle="border:1px
solid black">'+text+'</marquee>'
document.all.tooltip.style.pixelLeft=e.clientX+document.body.scrollLeft+
10
document.all.tooltip.style.pixelTop=e.clientY+document.body.scrollTop+10
document.all.tooltip.style.visibility="visible"
}
}

function hidetip(){
if (document.all)
document.all.tooltip.style.visibility="hidden"
}
//-->
</script>
```

此部分代码中有两个主要的函数 showtip(current,e,text)和 hidetip(), 它们分别是用来显示和隐藏提示条。在详细介绍它们如何工作之前, 先来看看页面的设置。

4. 把光盘内 sample\daohang\dhpmid.txt 中的“Part2”部分放入<body>与</body>标记之间。

```
<div id="tooltip" style="position:absolute;visibility:hidden;clip:rect(0
150 50 0);width:150px;background-color:seashell">

<table border="0">
<tr>
```

```
<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 字符从天而降!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>

<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 状态栏跑马灯 1!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>

<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 状态栏跑马灯 2!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>

<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 页面自动滚屏效果!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>
</tr>
<tr>
<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 隐藏显示字符脚本!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>
<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 滚动显示的 LOGO!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>
<td><font face="Arial, Helvetica, sans-serif" size="3"><b><a
href="..\blank\blank.htm" onMouseOver="showtip(this,event,' 鼠标跟踪器!') "
onMouseOut="hidetip()">新的链接</a></b></font></td>
</tr>
</table>
<hr>
```

这里，一开始放置了一个空的被标识为 tooltip 的<div></div>元素，它的可视属性 visibility 被设置为隐藏 hidden，这其实是为了放置提示条而设置的一个容器。起始它是不可见的，在后面可以通过函数在它内部放入提示条内容，并改变它的可视属性，使提示条显现出来。

通过链接的 onMouseOver 和 onMouseOut 事件分别触发对函数 showtip() 和 hidetip() 的调用，下面详细讲解这两个函数是如何实现其功能的。

showtip() 函数传递的 3 个参数中，this、event 和‘字符串’分别代表当前窗口、发生事件提示的内容。

一开始，函数 showtip() 作了一个判断：if (document.all&&document.readyState=="complete")，这主要是为了保证浏览器是 IE 以及页面已经全部载入。

接着，函数通过 document.all.tooltip 来检索出页面中放置的对象，这是一种有效而常用的方法。document.all 实际代表了页面中全部对象的集合，可以通过对象的标识在其中检索

单独的对象，如本例中的 tooltip。

innerHTML 属性是用来设置对象内部属性的 HTML 标签，这相当于把它的值（在这里是一条 <marquee></marquee> 语句）加入到对象标签中，这样就可以动态地修改提示条的内容。

然后，通过 `e.clientX+document.body.scrollLeft` 和 `e.clientY+document.body.scrollTop` 获得鼠标的位置，其中 `e.clientX` 和 `e.clientY` 表示鼠标指针在窗口可视范围中的坐标，而 `document.body.scrollLeft` 和 `document.body.scrollTop` 则分别表示当前窗口相对于文档原点的位移，这样相加的结果是得到了鼠标指针相对于整个文档的位置，在向右和向下各平移 10 个像素单位后，把它赋给 tooltip 对象的坐标属性。

最后，把 tooltip 对象的可视属性改为 visible 即可。

对于函数 `hidetip()`，则只是简单地在鼠标指针移出后把 tooltip 对象的可视属性重新设置为 hidden。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

第7篇 其他特效

案例51 设置打开窗口的特性

【效果说明】

本例中的程序可以用来定制具有个人特色的窗口，即打开链接时按照个人的喜好设置新的文档窗口的属性，包括是否打开工具栏的标准按钮、地址栏、链接栏，状态栏和菜单；能否自由调整窗口的大小，是否使用滚动条，设置窗口的宽和高等等。程序运行时的初始界面如图 51.1 所示。

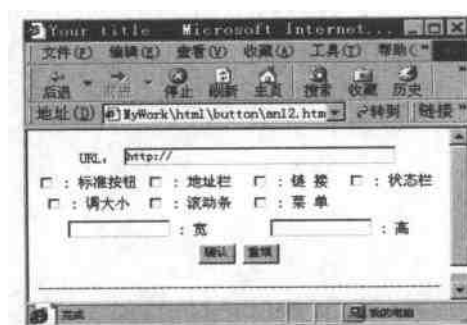


图 51.1 初始界面

在 URL 栏填入链接的地址，然后在下面的选择框中随意选取自己想要的属性，填入宽和高（如果这两项空白，则默认为上次打开的尺寸）。单击“确认”按钮后，程序就以设定的属性在新窗口打开文档，效果如图 51.2 所示。若单击“重填”按钮，则可以取消所有的选择。

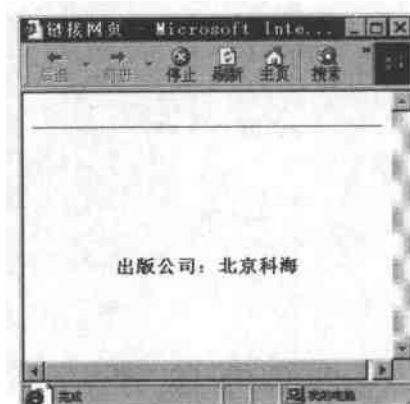


图 51.2 按照设定的属性打开文档窗口

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\other>windowattr.txt 中的“Part1”部分拷贝到<body>与</body>标记之间。

这段代码创建了一个表单 form1, 并且在这个表单中创建了 3 个文本框: 分别存储输入的链接地址、窗口的高度和宽度; 7 个复选框: 分别设置标准按钮、地址栏、链接、状态栏、滚动条、菜单和调大小的属性。最后定义了两个按钮“确认”和“重填”。值得注意的是, 这里的“确认”按钮和“重填”按钮, 虽然都是按钮, 但是它们的类型并不完全一样, 前者是普通按钮, 后者是复位按钮 reset, 代码如下:

```
<input type="button" size="10" value="确认" onClick="customize(this.form)" name="button">
```

```
<input type="reset" size="10" value="重填" name="reset">
```

普通按钮需要加入 onClick、onMouseDown、onMouseUp 等事件处理才有响应, 而单击“复位”按钮, 该表单的所有选项就全部清除, 不需加任何事件处理程序。

单击“确认”按钮后, 触发 onClick 事件处理程序, 调用 customize() 函数。

4. 将光盘内 sample\other>windowattr.txt 中的“Part2”部分复制到<head>与</head>标记之间。

这段程序包含一个重要的函数 customize()。此函数根据表单设置的结果定义了相应的特征变量, 代码如下:

```
var op_tool = (document.form1.tool.checked == true) ? 1 : 0;  
var op_loc_box = (document.form1.loc_box.checked == true) ? 1 : 0;  
var op_dir = (document.form1.dir.checked == true) ? 1 : 0;  
var op_stat = (document.form1.stat.checked == true) ? 1 : 0;  
var op_menu = (document.form1.menu.checked == true) ? 1 : 0;  
var op_scroll = (document.form1.scroll.checked == true) ? 1 : 0;  
var op_resize = (document.form1.resize.checked == true) ? 1 : 0;
```

例如, 为了存放“标准按钮”此项的属性, 函数定义了 op_tool 变量, 并根据表单中对此项的设置为该变量赋值, 代码如下:

```
var op_tool = (document.form1.tool.checked == true) ? 1 : 0
```

上面一行代码的含义为: 如果表单中此项被选中(document.form1.tool.checked == true), op_tool 的值就设置为 1, 否则就设置为 0。

此外，函数把链接的地址项 URL 的值存入 `address` 变量中，把窗口的高度和宽度分别存放在变量 `op_wid` 和 `op_height` 中，代码如下：

```
var address = document.form1.url.value;
var op_wid  = document.form1.wid.value;
var op_heigh = document.form1.heigh.value;
```

然后把这些属性连接成字符串 `option`，代码如下：

```
var option="toolbar="+op_tool+",location="+op_loc_box+",directories="
+ op_dir + ",status=" + op_stat + ",menubar=" + op_menu + ",scrollbars="
+ op_scroll + ",resizeable=" + op_resize + ",width=" + op_wid
+ ",height=" + op_heigh;
```

最后调用 `window.open()` 函数按照设置打开一个新的窗口 `what_I_want`，并且把链接文档在这个窗口中打开，代码如下：

```
window.open("", "what_I_want", option);
window.open(address, "what_I_want");
```

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例52 日 历

【效果说明】

本例是一个日程序。在浏览器中运行此程序，在页面中就会根据本地计算机当前时间显示出年月日。如图 52.1 所示。

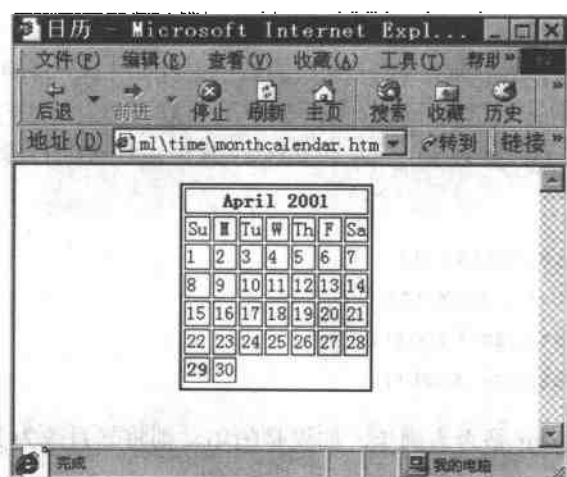


图 52.1 日历

如果把计算机的时间改为 1999 年 3 月 29 日，然后单击浏览器工具栏中的“刷新”按钮，则程序的运行结果如图 52.2 所示。

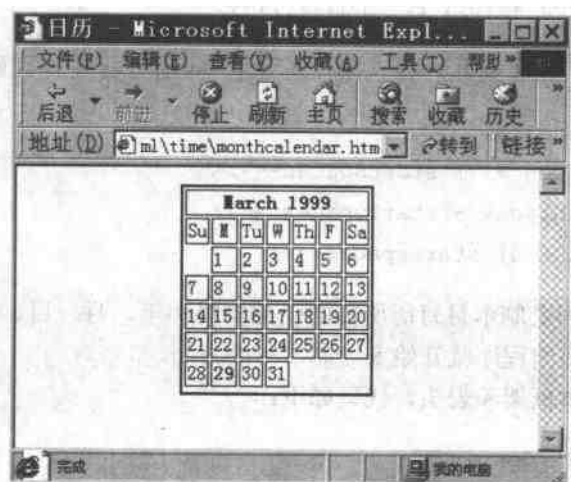


图 52.2 改变计算机时间后程序的运行结果

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 将光盘内 sample\other\monthcalendar.txt 中的全部内容拷贝到<head>与</head>标记之间。

此段程序首先创建了一个拥有 12 个元素的数组来存储 12 个月的名字, 然后创建了一个数组来存储 12 个月一般情况下每个月所拥有的天数。

分别用 **thismonth**、**thisdate**、**thisyear** 和 **thisday** 来记录现在的月份、日期、年份和星期几, 代码如下:

```
thisday=todayDate.getDay();
thismonth=todayDate.getMonth();
thisdate=todayDate.getDate();
thisyear=todayDate.getFullYear();
```

然后判断今年 **thisyear** 是否为闰年。如果是闰年, 则将二月改为 29 天, 代码如下:

```
thisyear = thisyear % 100;
thisyear = ((thisyear < 50) ? (2000 + thisyear) : (1900 + thisyear));
if (((thisyear % 4 == 0)
&& !(thisyear % 100 == 0))
|| (thisyear % 400 == 0)) monthdays[1]++;
```

计算本月的第一天是星期几, 将它保存在变量 **startspaces** 中, 代码如下:

```
startspaces=thisdate;
while (startspaces > 7) startspaces-=7;
startspaces = thisday - startspaces + 1;
if (startspaces < 0) startspaces+=7;
```

到这里, 已经得到绘制本月月历所需的全部资料: 年、月、日、本月有多少天, 第一天从星期几开始。下面的程序就开始来绘制本月的月历。

程序先绘制了总体框架和表头, 代码如下:

```
document.write("<Center>");
document.write("<table border=2 bgcolor=#ffffff ");
document.write("bordercolor=black><font color=black>");
document.write("<tr><td colspan=7><center><strong>"
+ monthnames[thismonth] + " " + thisyear
```

```
+ "</strong></center></font></td></tr>");
document.write("<tr>");
document.write("<td align=center>Su</td>");
document.write("<td align=center>M</td>");
document.write("<td align=center>Tu</td>");
document.write("<td align=center>W</td>");
document.write("<td align=center>Th</td>");
document.write("<td align=center>F</td>");
document.write("<td align=center>Sa</td>");
document.write("</tr>");
```

由于本月的第一天的是星期 `startspaces`，因此需要把星期一到星期 `startspaces-1` 这几格空起来，代码如下：

```
for (s=0;s<startspaces;s++)
{
    document.write("<td> </td>");
}
```

将日期 `count` 赋为 1，从 1 到 `monthdays[thismonth]` 将数字逐个填入表中，代码如下：

```
count=1;
while (count <= monthdays[thismonth])
{
    for (b = startspaces;b<7;b++)
    {
        document.write("<td>");
        if (count <= monthdays[thismonth])
        {
            document.write(count);
        }
        else {
            document.write(" ");
        }
        document.write("</td>");
        count++;
    }
    document.write("</tr>");
    document.write("<tr>");
    startspaces=0;
}
```

为了使表示“今天”（即当前日期）的数字醒目一些，特将代表“今天”的数字用红色表示，并将它加粗，代码如下：

```
if (count==thisdate)
{
    document.write("<font color='FF0000'><strong>");
}
if (count==thisdate)
{
    document.write("</strong></font>");
}
```

4. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。

案例53 追踪来访次数

从现在开始，将要讲解如何用 JavaScript 程序来设置和使用 Cookie。在讲解具体的例子之前，先来学习一些 Cookie 的基本知识。

这里所说的 Cookie 是指保存在用户机器上，用于存储用户系统信息的对象，通过它，可以在用户的机器上保存一些有用的信息。当用户打开页面时，页面中的程序就通过读取 Cookie 中的内容来获取这些信息。通常，这些信息包括诸如用户姓名、ID、密码等内容。

一个 Cookie 包含有 6 个主要元素：name、value、expires、domain、path 和 secure，它们决定了 Cookie 的作用范围和存储的数据，下面来具体看看这些元素。

name: Cookie 的名字，用来识别 Cookie 的标识。name 元素是一个不含分号，逗号和空格的字符串，可以像命名变量一样命名一个 Cookie。name 元素是必不可少的。

value: Cookie 的值，是实际存放于 Cookie 中的信息，它是由任意字符构成的字符串。value 元素也是必需的。

expires: Cookie 的过期时间，它决定 Cookie 过期的时间。过期并不意味着 Cookie 消失，它只是不再被发送，在它的位置被覆盖之前，它依然存在。一个没有设置 expires 的 Cookie 在用户断开连接后就过期。

domain: Cookie 所在的域名，它指出此 Cookie 所在的域名。这样，不同站点间的 Cookie 不会相互影响。只有拥有该域名的站点才能为那个域名设置 Cookie。

path: 一个 Cookie 可以被指定为只针对一个站点的某一层级。path 是可选项，如果没有指定 path，将被缺省地设置为 Cookie 页面的路径。

secure: secure 标志是一个布尔值，当它为真时，此 Cookie 只对被用户浏览器认为是安全的服务器可用；为假时，则对所有服务器可用。secure 的默认值是假。

注意：一个 Cookie 中可以保存多组元素的集合，而在提取 Cookie 时，它们一起被返回，这时需要分析返回的字符串来获取所要的值，这在后面分析程序时会讲解如何处理。

在介绍过一些 Cookie 的基本知识以后，下面就来看一个具体的例子。此例通过使用 Cookie，实现对用户访问次数的统计。在此例中，涉及到了 Cookie 的设置、修改、访问等各个方面，而且它使用的方法也具有普遍意义。因此，如果大家看懂了此例，那么对 Cookie 的使用也就基本掌握了。

【效果说明】

在如图 53.1 所示页面中显示了欢迎词和来过此页面的次数。当刷新页面或是重新打开此页面，就会发现欢迎词有了变化，并且可以看到浏览此页面的次数也增加了，如图 53.2 所示。重复以上操作，欢迎词还会变化，直到第 5 次以后，欢迎词才不再变化，但对访问次数的统计仍然会继续增长，如图 53.3~图 53.5 所示。

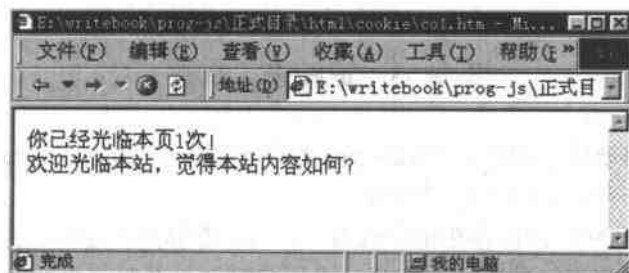


图 53.1 第1次访问页面的欢迎词

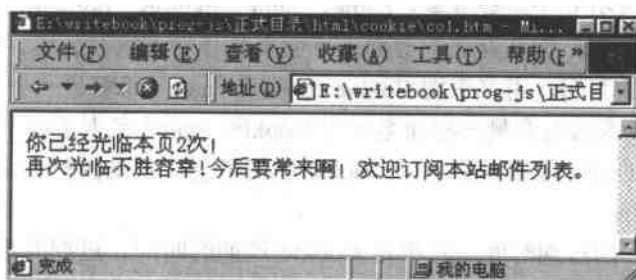


图 53.2 第2次访问页面的欢迎词

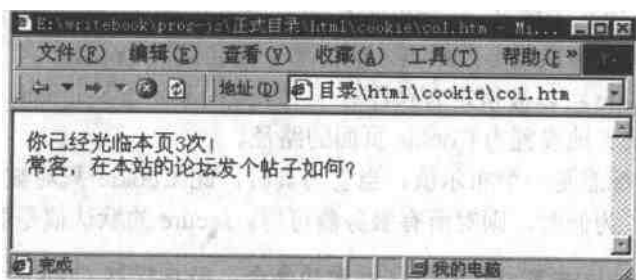


图 53.3 第3次访问页面的欢迎词

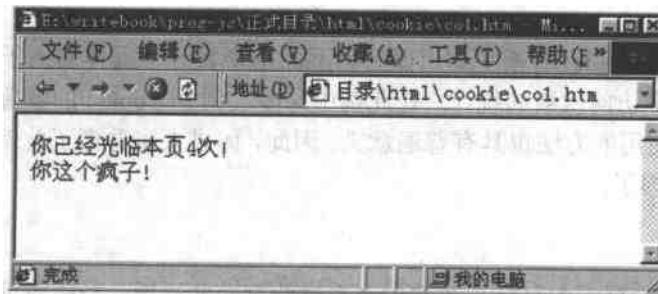


图 53.4 第4次访问页面的欢迎词

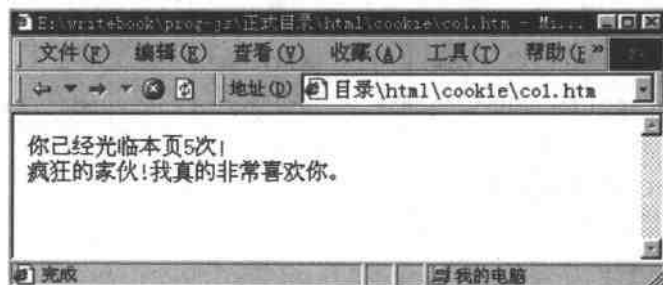


图 53.5 第 5 次访问页面的欢迎词

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000，并确定已进入“网页”视图（选择“查看”菜单下的“网页”命令）。
2. 单击窗口底部的 HTML 标签，切换到 HTML 源文件编辑模式。
3. 先来看看程序中相关函数的内容，将光盘内 sample\cookie\count.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

```
function SetCookie(name,value)
{
    var argv=SetCookie.arguments;
    var argc=SetCookie.arguments.length;
    var expires=(2<argc)?argv[2]:null;
    var path=(3<argc)?argv[3]:null;
    var domain=(4<argc)?argv[4]:null;
    var secure=(5<argc)?argv[5]:false;
    document.cookie=name+"="+escape(value)+((expires==null)?"":(";expires="
+expires.toGMTString()))+((path==null)?"":(";path="+path))+
((domain==null)?"":(";domain="+domain))+((secure==true)?"; secure":""));
}
```

函数 SetCookie(name,value)用于设置 Cookie，参数 name 和 value 是必需的，在使用时也可以加入其他参数。变量 argv 和 argc 分别表示此函数被调用时的参数值和参数长度。对参数进行处理，在有相应参数的情况下，把它们的值分别赋给 expires、path、domain 和 secure 变量；否则，把这些变量设为 null 或 false。此函数具有通用性，可以用它来设置你的 Cookie。注意，在使用 document.cookie 设置 Cookie 值的时候，可以看到，它是以“name=value; expires; path; domain; secure”的形式保存的，这在以后进行读取 Cookie 操作时需要用到。另外，value 值是经过 escape 方法重新编码过的，以便使它的值在各种类型的计算机上都能被读取。

```
function getCookieVal(offset)
{
```

```
var endstr=document.cookie.indexOf(";",offset);
if(endstr==-1)endstr=document.cookie.length;
return unescape(document.cookie.substring(offset,endstr));)

function GetCookie(name)
{
    var arg=name+"=";
    var alen=arg.length;
    var clen=document.cookie.length;
    var i=0;
    while(i<clen){
        var j=i+alen;
        if(document.cookie.substring(i,j)==arg)
            return getCookieVal(j);
        i=document.cookie.indexOf(" ",i)+1;
        if(i==0)break;
    }
    return null;}

```

函数 `GetCookie(name)` 和 `getCookieVal(offset)` 用来获取所需的 Cookie 值。前面已经提到过，当获取当前页面的 Cookie 时，用 `document.cookie` 的返回值中可能包含有多组值，对此需要进行分析，只提取想要的内容。

先来看看函数 `GetCookie(name)` 的工作过程。变量 `arg` 的值是要搜索的名字再加上一个“=”，此字符串变量是用来搜索的依据，这是因为，名字是独一无二的。变量 `alen` 和 `clen` 分别代表 `arg` 和 Cookie 的长度。然后，在 Cookie 长度范围内进行搜索：当找到与 `arg` 匹配的名字的时候，也就是 `document.cookie.substring(i,j) = arg` 时，通过调用函数 `getCookieVal(j)` 来返回所需的值；否则，`i=document.cookie.indexOf(" ",i)+1`，也就是跳到下一组值的开始处来进行搜索。这里可以看出，在 Cookie 中，每组值之间是通过空格来分隔的。如果 `i = 0`，代表 Cookie 中没有内容，跳出循环，返回空值。

再来看看函数 `getCookieVal(offset)` 是如何得到所需的 value 值的。变量 `endstr = document.cookie.indexOf(";",offset)`，方法 `indexOf` 返回从 `offset` 处开始，到第一个“;”处的字符串长度，这就是所需的值；如果 `endstr = -1`，则说明没有找到“;”，也就是说，这个 Cookie 里只有这一组值，且除了 `name = value` 以外，其他的各项都没有设置。因此，只要令 `endstr = document.cookie.length`，就可以了。然后，返回值 `unescape(document.cookie.substring(offset,endstr))`，这里，先通过 `substring(offset,endstr)` 方法获得 value 的值，然后用 `unescape` 方法将它解码还原成 ASCII 字符。

这样，就可以得到想要的值。

4. 下面看看如何通过以上函数来实现页面效果，将光盘内 `sample\cookie\count.txt` 中的“Part2”部分拷贝到 `<body>` 与 `</body>` 标记之间。

```
<script language="JavaScript">
<!--
var expdate=new Date();
var visits;

expdate.setTime(expdate.getTime()+(24*60*60*1000*365));

if(!(visits=GetCookie("visits"))visits=0;
visits++;
SetCookie("visits",visits,expdate,"/",null,false);

document.write("你已经光临本页"+"<FONT COLOR=red>" +visits+"</font>"+"次!");

if(visits==1) document.write("<br>"+"欢迎光临本站, 觉得本站内容如何?");

if(visits==2) document.write("<br>"+"再次光临不胜荣幸!今后要常来啊!
                        欢迎订阅本站邮件列表。");

if(visits==3) document.write("<br>"+"常客, 在本站的论坛发个帖子如何?");

if(visits==4) document.write("<br>"+"你这个疯子!");

if(visits>=5) document.write("<br>"+"疯狂的家伙!我真的非常喜欢你。");
//-->
</script>
```

这里, 先声明一个代表 Cookie 过期时间的变量 expdate 和一个用来代表访问次数的变量 visits, 然后通过 expdate.setTime(expdate.getTime()+(24*60*60*1000*365)) 把 expdate 设为当前时间一年后。这里, 此设置是可以更改的, 可把过期时间改变为所需的值。接着, 当 visits=GetCookie("visits") 返回值为假, 也就是说以前没有设置过这个名为 visits 的 Cookie 的时候, 为 visits 变量赋初值 0。然后, visits 值加 1, 重新设置 Cookie, 把改变后的 visits 保存下来。最后, 通过对不同的 visits 值进行区分, 在页面上书写不同的欢迎词和统计次数。

5. 将文件保存为 HTML 格式, 并在 IE 浏览器中打开就可以看到前面所述的效果。

案例54 记录上次访问时间

【效果说明】

本例中的程序可以记录下访问页面的用户姓名，访问的次数和上次访问的时间。如果用户是第一次访问此页面，程序弹出一个对话框，要求用户输入姓名，如图 54.1 所示。这里，程序默认的用户名是 Guest，如果选择了取消或是默认的用户名，在接下来的页面里，会出现相应的提示，如图 54.2 和图 54.3 所示。这两种情况下，程序都不会记录访问次数和访问时间。

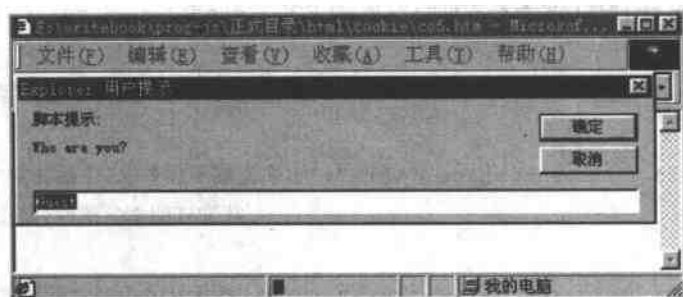


图 54.1 第 1 次访问页面弹出对话框

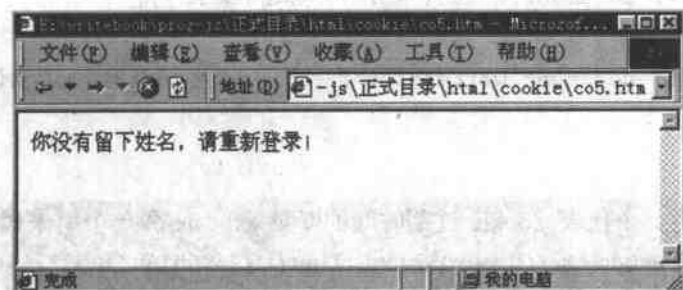


图 54.2 选择取消后出现的提示

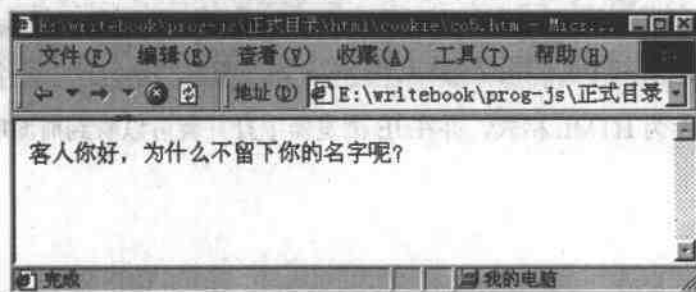


图 54.3 选择默认用户名后出现的提示

如果用户留下了姓名，程序将记录用户的名字，并且会统计用户的访问次数和上次访

问时间, 如图 54.4 和图 54.5 所示。

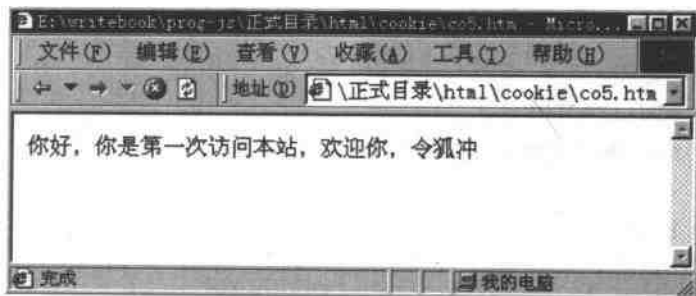


图 54.4 输入用户名后的页面

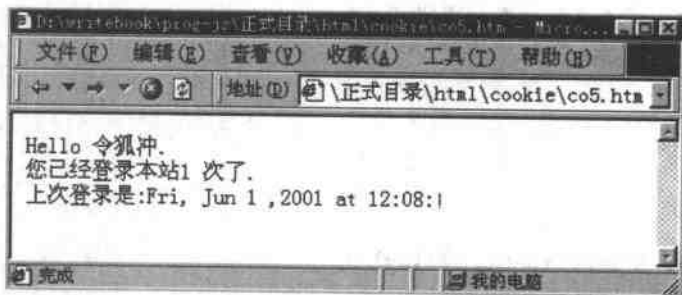


图 54.5 程序记录了用户名、访问次数和上次访问时间

【创作步骤与关键点解析】

下面是制作此页面的步骤。

1. 启动 FrontPage 2000, 并确定已进入“网页”视图(选择“查看”菜单下的“网页”命令)。
2. 单击窗口底部的 HTML 标签, 切换到 HTML 源文件编辑模式。
3. 先来看看程序中相关函数的内容, 将光盘内 sample\cookie\lastVisit.txt 中的“Part1”部分拷贝到<head>与</head>标记之间。

```
var expDays = 30;  
var exp = new Date();  
exp.setTime(exp.getTime() + (expDays*24*60*60*1000));
```

此段代码求出了 Cookie 的过期时间, 并把它保存在变量 exp 中。这里时间是 30 天, 也可以通过自行修改来选择合适的。

```
function getCookieVal (offset) {  
var endstr = document.cookie.indexOf (";", offset);  
if (endstr == -1)  
endstr = document.cookie.length;  
return unescape(document.cookie.substring(offset, endstr));
```

```
}

function GetCookie (name) {
    var arg = name + "=";
    var alen = arg.length;
    var clen = document.cookie.length;
    var i = 0;
    while (i < clen) {
        var j = i + alen;
        if (document.cookie.substring(i, j) == arg)
            return getCookieVal (j);
        i = document.cookie.indexOf(" ", i) + 1;
        if (i == 0) break;
    }
    return null;
}

function SetCookie (name, value) {
    var argv = SetCookie.arguments;
    var argc = SetCookie.arguments.length;
    var expires = (argc > 2) ? argv[2] : null;
    var path = (argc > 3) ? argv[3] : null;
    var domain = (argc > 4) ? argv[4] : null;
    var secure = (argc > 5) ? argv[5] : false;
    document.cookie = name + "=" + escape (value) +
        ((expires == null) ? "" : ("; expires=" + expires.toGMTString())) +
        ((path == null) ? "" : ("; path=" + path)) +
        ((domain == null) ? "" : ("; domain=" + domain)) +
        ((secure == true) ? "; secure" : "");
}

function DeleteCookie (name) {
    var exp = new Date();
    exp.setTime (exp.getTime() - 1);
    var cval = GetCookie (name);
    document.cookie = name + "=" + cval + "; expires=" + exp.toGMTString();
}
```

函数 `SetCookie (name, value)`, `GetCookie (name)`和 `getCookieVal (offset)`跟案例 53 中讲到的基本相同, 就不再介绍了。这里增加了一个函数 `DeleteCookie (name)`, 可以看到, 这个函数是通过把 Cookie 的过期时间设为当前系统时间来实现的。

```
function Who(info){
var VisitorName = GetCookie('VisitorName')
if ((VisitorName == null)|| (VisitorName == 'Guest')) {
VisitorName = prompt("Who are you?", "Guest");
DeleteCookie("VisitorName");
DeleteCookie("WWHCount");
DeleteCookie("WWhenH");
if(VisitorName != null)SetCookie ('VisitorName', VisitorName, exp);
else return null;
}
return VisitorName;
}
```

函数 `Who()`首先检查用户是否已经访问过网页并保存有 Cookie, 如果有的话, 就返回 Cookie 中保存的用户名; 如果没有的话, 就通过弹出一个对话框来要求用户输入姓名。程序对对话框返回的值做分析: 对于空的用户名, 程序返回空值; 否则, 程序设置一个 Cookie 保存用户名, 然后返回此用户名。注意, 这里是根据客户端计算机上的 Cookie 来识别用户的。所以, 如果不同的用户使用同一台计算机登录此页面, 看到的用户名将是第 1 个登录的用户名。

```
function When(info){
var rightNow = new Date()
var WWHTime = 0;
WWHTime = GetCookie('WWhenH')
WWHTime = WWHTime * 1
var lastHereFormatting = new Date(WWHTime);
var lastHereInDateFormat = "" + lastHereFormatting;
var dayOfWeek = lastHereInDateFormat.substring(0,3)
var dateMonth = lastHereInDateFormat.substring(4,10)
var timeOfDay = lastHereInDateFormat.substring(10,16)
var year = lastHereInDateFormat.substring(28,33)
var WWHText = dayOfWeek + ", " + dateMonth + ", " + year + " at " + timeOfDay
SetCookie ("WWhenH", rightNow.getTime(), exp)
return WWHText
}
```

函数 When()先声明一个时间变量 rightNow, 显然, 它用来保存当前的时间。然后, 变量 WWHTime 用来保存从 Cookie “WWHenH” 中获取的上次访问时间, 这里 WWHTime 是一个字符串, 通过 WWHTime = WWHTime * 1 操作, 使它变成整数变量。然后, 用 WWHTime 做初值, 将它赋给一个时间变量 lastHereFormatting, 这个时间变量是用毫秒数表示的上次访问时间与 1970 年 1 月 1 日之间的差距。通过 lastHereInDateFormat = "" + lastHereFormatting;操作, 将时间变量 lastHereFormatting 转化成用字符串表示的时间变量, 并赋给变量 lastHereInDateFormat 保存。这样, 可以提取此变量中的相应部分来获得上次访问是星期几及月份、日期、年份和时间。然后, 把这些值组成一个字符串返回, 在此之前, 需要把现在的时间写入 Cookie。

```
function Count(info){
var WWHCount = GetCookie('WWHCount')
if (WWHCount == null) {
WWHCount = 0;
}
else{
WWHCount++;
}
SetCookie ('WWHCount', WWHCount, exp);
return WWHCount;
}
```

函数 Count()用来统计用户访问网页的次数, 此函数与案例 53 中的统计次数的函数结构基本相同, 就不再赘述。

4. 下面看看如何通过以上函数来实现页面效果, 将光盘内 sample\cookie\lastVisit.txt 中的 “Part2” 部分拷贝到<body>与</body>标记之间。

```
<script language="JavaScript">
<!--
var visitorName = Who();
var visitCount = Count();
var lastVisitTime = When();
if(visitorName == null)
document.write("你没有留下姓名, 请重新登录!");
else if(visitorName == 'Guest')
    document.write("客人你好, 为什么不留下你的名字呢?");
else if(visitCount == 0)
    document.write("你好, 你是第一次访问本站, 欢迎你, " + visitorName);
else
    { document.write("Hello " + visitorName);
```

```
document.write(" <br>您已经登录本站" + visitCount + " 次了.  
                <br>上次登录是:" + lastVisitTime + "! ");  
-->  
</script>
```

可以看到，此段程序主要是通过调用函数来获取 Cookie 中的值，并对返回值做一些判断，主要是针对没有留下姓名或采用了默认用户名的情况来改变页面显示的内容。而当用户第一次访问页面时，由于并没有上次的访问时间，因此也需要有针对性地改变页面内容。

5. 将文件保存为 HTML 格式，并在 IE 浏览器中打开就可以看到前面所述的效果。