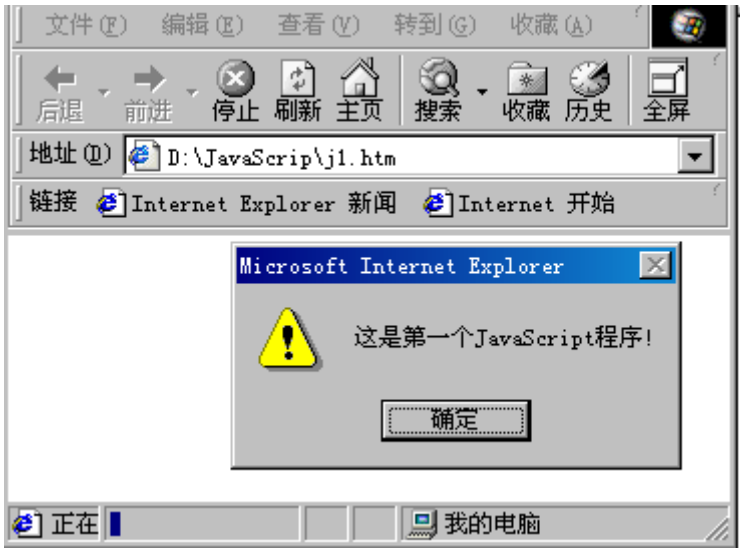
JavaScript 教程

一、 编写第一个 JavaScript 程序

下面我们通过一个例子，编写第一个 JavaScript 程序。通过它可说明 JavaScript 的脚本是怎样被嵌入到 HTML 文档中的。test1.html 文档：

```
<html>
<head>
<Script Type ="JavaScript">
// JavaScript Appears here.
alert("这是第一个 JavaScript 例子!");
alert("欢迎你进入 JavaScript 世界!");
alert("今后我们将共同学习 JavaScript 知识! ");
</Script>
</Head>
</Html>
```

在 Internet Explore5.0 中运行行后的结果见图 1－1 所示。



说明： test.html 是 HTML 文档，其标识格式为标准的 HTML 格式；如同 HTML 标识语言一样， JavaScript 程序代码是一些可用字处理软件浏览的文本，它在描述页面的 HTML 相关区域出现。JavaScript 代码由 <Script Language ="JavaScript">...</Script> 说明。在标识<Script Language ="JavaScript">...</Script>之间就可加入 JavaScript 脚本。

alert()是 JavaScript 的窗口对象方法,其功能是弹出一个具有 OK 对话框并显示()中的字符串。通过<!--...//--> 标识说明：若不认识 JavaScript 代码的浏览器，则所有在其中的标识均被忽略；若认识，则执行其结果。使用注释这是一个好的编程习惯，它使其他人可以读懂你的语言。JavaScript 以 </Script> 标签结束。

从上面的实例分析中我们可以看出，编写一个 JavaScript 程序确实非常容易的。

二、JavaScript 基本数据结构

JavaScript 提供脚本语言的编程与 C++非常相似，它只是去掉了 C 语言中有关指针等容易产生的错误，并提供了功能强大的类库。对于已经具备 C ++或 C 语言的人来说，学习 JavaScript 脚本语言是一件非常轻松愉快的事。

1. JavaScript 代码的加入

JavaScript 的脚本包括在 HTML 中，它成为 HTML 文档的一部分。与 HTML 标识相结合，构成了一个功能强大的 Internet 网上编程语言。可以直接将 JavaScript 脚本加入文档：

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

说明：

通过标识<Script>...</Script>指明 JavaScript 脚本源代码将放入其间。

通过属性 Language ="JavaScript"说明标识中是使用的何种语言，这里是 JavaScript 语言，表示在 JavaScript 中使用的语言。

下面是将 JavaScript 脚本加入 Web 文档中的例子:

Test2.html

```
<HTML>
<Head>
<Script Language ="JavaScript">
document. Write("这是赛迪网互动学校");
document. close();
</Script>
</Head>
</HTML>
```

在浏览器的窗口中调用 test2.html，则显示“这是赛迪网互动学校”字串。见图 2 所示。

说明:

Document. write()是文档对象的输出函数，其功能是将括号中的字符或变量值输出到窗口；document. close()是将输出关闭。

可将<Script>...</Script>标识放入 head>.. </Head>或<Body> ...</Body>之间。将 JavaScript 标识放置<Head>... </Head>在头部之间，使之在主页和其余部分代码之前装载，从而可使代码的功能更强大；可以将 JavaScript 标识放置在<Body>... </Body>主体之间以实现某些部分动态地创建文档。

图 2

2. 基本数据类型

JavaScript 脚本语言同其它语言一样，有它自身的基本数据类型、表达式和算术运算符以及程序的基本框架结构。JavaScript 提供了四种基本的数据类型用来处理数字和文字，而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。

(1) 基本数据类型

在 JavaScript 中四种基本的数据类型：数值（整数和实数）、字符串型（用“”号或‘’括起来的字符或数值）、布尔型（使 True 或 False 表示）和空值。在 JavaScript 的基本类型中的数据可以是常量，也可以变量。由于

JavaScript 采用弱类型的形式，因而一个数据的变量或常量不必首先作声明，而是在使用或赋值时确定其数据的类型的。当然也可以先声明该数据的类型，它是通过在赋值时自动说明其数据类型的。

(2) 常量

(整型常量)-JavaScript 的常量通常又称字面常量，它是不能改变的数据。其整型常量可以使用十六进制、八进制和十进制表示其值。

(实型常量)-实型常量是由整数部分加小数部分表示，如 12.32、193.98 。可以使用科学或标准方法表示：5E7、4e5 等。

(布尔值)-布尔常量只有两种状态：True 或 False。 它主要用来说明或代表一种状态或标志，以说明操作流程。它与 C ＋＋是不一样的, C ＋＋可以用 1 或 0 表示其状态,而 JavaScript 只能用 True 或 False 表示其状态。

(字符型常量)-使用单引号（ ‘）或双引号（ “）括起来的一个或几个字符。如 ″This is a book of JavaScript ″、″3245″、″ewrt234234″ 等。

(空值)-JavaScript 中有一个空值 null,表示什么也没有。如试图引用没有定义的变量,则返回一个 Null 值。

(特殊字符)-同 C 语言一样，JavaScript 中同样以有些以反斜杠（ / ）开头的不可显示的特殊字符。通常称为控制字符。

(3) 变量

变量的主要作用是存取数据、提供存放信息的容器。对于变量必须明确变量的命名、变量的类型、变量的声明及其变量的作用域。

变量的命名

JavaScript 中的变量命名同其计算机语言非常相似，这里要注意以下两点：

Ａ、必须是一个有效的变量，即变量以字母开头，中间可以出现数字如 test1、text2 等。除下划线（－）作为连字符外，变量名称不能有空格、（＋）、（－）、（，）或其它符号。

Ｂ、不能使用 JavaScript 中的关键字作为变量。

在 JavaScript 中定义了 4 0 多个类键字，这些关键是 JavaScript 内部使用的，不能作为变量的名称。如 Var、int、double、true 不能作为变量的名称。

在对变量命名时，最好把变量的意义与其代表的意思对应起来，以免出现错误。

变量的类型

在 JavaScript 中，变量可以用命令 Var 作声明:

var mytest;

该例子定义了一个 mytest 变量。但没有赋予它的值。

Var mytest=”This is a book”

该例子定义了一个 mytest 变量, 同时赋予了它的值。

在 JavaScript 中，变量以可以不作声明，而在使用时再根据数据的类型来确其变量的类型。

如：

x=100　　y=″125″　　xy= True　　cost=19.5 等。其中 x 整数，y 为字符串，xy 为布尔型，cost 为实型。

变量的声明及其作用域

JavaScript 变量可以在使用前先作声明，并可赋值。通过使用 var 关键字对变量作声明。对变量作声明的最大好处就是能及时发现代码中的错误；因为 JavaScript 是采用动态编译的，而动态编译是不易发现代码中的错误，特别是变量命名的方面。

对于变量还有一个重要性——那就是变量的作用域。在 JavaScript 中同样有全局变量和局部变量。全局变量是定义在所有函数体之外，其作用范围是整个函数；而局部变量是定义在函数体之内，只对其该函数是可见的，而对其它函数则是不可见的。

3.表达式和运算符

(1) 表达式

在定义完变量后，就可以对它们进行赋值、改变、计算等一系列操作，这一过程通常又叫称一个叫表达式来完成，可以说它是变量、常量、布尔及运算符的集合，因此表达式可以分为算术表述式、字串表达式、赋值表达式以及布尔表达式等。

(2) 运算符

运算符完成操作的一系列符号，在 JavaScript 中有算术运算符，如＋、-、*、/等；有比较运算符如!＝、＝＝等；有逻辑布尔运算符如！（取反）、|、||；有字串运算如＋ 、 ＋＝等。

在 JavaScript 主要有双目运算符和单目运算符。其双目运算符由下列组成：

操作数 1 运算符 操作数 2

即由两个操作数和一个运算符组成。如 50＋40、"This"+"that"等。单目运算符，只需一个操作数，其运算符可在前或后。

（1）算术运算符

JavaScript 中的算术运算符有单目运算符和双目运算符。

双目运算符：

+（加） 、-（减）、 *（乘）、 /（除）、 %（取模） 、|（按位或）、&(按位与)、<<（左移）、 >>（右移）、>>>（右移，零填充）。

单目运算符：

-（取反）、~（取补）、++（递加 1）、--（递减 1）。

（2）比较运算符

比较运算符它的基本操作过程是，首先对它的操作数进行比较，尔后再返回一个 true 或 False 值，有 8 个比较运算符:

<(小于)、>(大于)、<=(小于等于)、>=(大于等于)、==(等于)、!=(不等于)。

（3）布尔逻辑运算符

在 JavaScript 中增加了几个布尔逻辑运算符:

!（取反)、&=（与之后赋值）、 &（逻辑与）、 |=（或之后赋值）、 |（逻辑或）、 ^=（异或之后赋值）、 ^（逻辑异或）、 ?:（三目操作符）、||（或）、==(等于)、|=(不等于)。

其中三目操作符主要格式如下：

操作数？结果 1：结果 2

若操作数的结果为真，则表述式的结果为结果 1，否则为结果 2。

4.范例

下面是一个跑马灯效果的 JavaScript 文档。

Test2_1.html

```
<html>
<head>
<script Language="JavaScript">
var msg="这是一个跑马灯效果的 JavaScript 文档";
var interval = 100;
var spacelen = 120;
var space10="  ";
var seq=0;
function Scroll() {
len = msg.length;
window.status = msg.substring(0, seq+1);
seq++;
if ( seq >= len ) {
seq = spacelen;
```

```
window.setTimeout("Scroll2();", interval );
}
else
window.setTimeout("Scroll();", interval );
}
function Scroll2() {
var out="";
for (i=1; i<=spacelen/space10.length; i++) out +=
space10;
out = out + msg;
len=out.length;
window.status=out.substring(seq, len);
seq++;
if ( seq >= len ) { seq = 0; };
window.setTimeout("Scroll2();", interval );
}
Scroll();
</script>
<body>
</body>
</html>
```

本文介绍了 JavaScript 脚本是如何加入 Web 页面，并学习了 JavaScript 语言中的基本数据类型、变量、常量、操作运算符等。可以看出，对于已经掌握 C ＋＋语言的人来说，学习 JavaScript 真是一件非常轻松愉快的事。

三、JavaScript 程序构成

JavaScript 脚本语言的基本构成是由控制语句、函数、对象、方法、属性等,来实现编程的。

1. 程序控制流

在任何一种语言中，程序控制流是必须的，它能使得整个程序减小混乱，使之顺利按其一定的方式执行。下面是 JavaScript 常用的程序控制流结构及语句:

(1) if 条件语句

基本格式

if（表述式）

语句段 1；

……

Else

语句段 2；

……

功能：若表达式为 true，则执行语句段 1；否则执行语句段 2。

说明：

if -else 语句是 JavaScript 中最基本的控制语句，通过它可以改变语句的执行顺序。

表达式中必须使用关系语句，来实现判断，它是作为一个布尔值来估算的。

它将零和非零的数分别转化成 false 和 true。

若 if 后的语句有多行，则必须使用花括号将其括起来。

if 语句的嵌套

if（布尔值）语句 1；

else（布尔值）语句 2；
else if（布尔值）语句 3；
……
else 语句 4；
在这种情况下，每一级的布尔表述式都会被计算，若为真，则执行其相应的语句，否则执行 else 后的语句。
(2)For 循环语句
基本格式
for（初始化；条件；增量）
语句集；
功能：实现条件循环，当条件成立时，执行语句集，否则跳出循环体。
说明：初始化参数告诉循环的开始位置，必须赋予变量的初值；
条件：是用于判别循环停止时的条件。若条件满足，则执行循环体，否则 跳出。
增量：主要定义循环控制变量在每次循环时按什么方式变化。
三个主要语句之间，必须使用逗号分隔。
(3) while 循环
基本格式
while（条件）
语句集；
该语句与 For 语句一样，当条件为真时，重复循环，否则退出循环。
For 与 while 语句
两种语句都是循环语句，使用 For 语句在处理有关数字时更易看懂，也较紧凑；而 while 循环对复杂的语句效果更特别。
(4) break 和 continue 语句　　与 C++语言相同，使用 break 语句使得循环从 For 或 while 中跳出，continue 使得跳过循环内剩余的语句而进入下一次循环。

2. 函数

函数为程序设计人员提供了一个丰常方便的能力。通常在进行一个复杂的程序设计时，总是根据所要完成的功能，将程序划分为一些相对独立的部分，每部分编写一个函数。从而，使各部分充分独立，任务单一，程序清晰，易懂、易读、易维护。JavaScript 函数可以封装那些在程序中可能要多次用到的模块。并可作为事件驱动的结果而调用的程序。从而实现一个函数把它与事件驱动相关联。这是与其它语言不样的地方。
(1) JavaScript 函数定义
Function 函数名 （参数,变元）{
函数体;
Return 表达式;
}
说明：
当调用函数时,所用变量或字面量均可作为变元传递。
函数由关键字 Function 定义。
函数名：定义自己函数的名字。
参数表，是传递给函数使用或操作的值，其值可以是常量 ，变量或其它表达式。
通过指定函数名（实参）来调用一个函数。
必须使用 Return 将值返回。
函数名对大小写是敏感的。
(2) 函数中的形式参数：
在函数的定义中，我们看到函数名后有参数表，这些参数变量可能是一个或几个。那么怎样才能确定参数变量的个数呢？在 JavaScript 中可通过 arguments .Length 来检查参数的个数。例：

```
Function function_Name(exp1,exp2,exp3,exp4)
Number =function _Name . arguments .length;
if (Number>1)
document.wrile(exp2);
if (Number>2)
document.write(exp3);
if(Number>3)
document.write(exp4);
...
```

3.事件驱动及事件处理

(1)基本概念

JavaScript 是基于对象(object-based)的语言。这与 Java 不同,Java 是面向对象的语言。而基于对象的基本特征，就是采用事件驱动(event-driven)。它是在用形界面的环境下，使得一切输入变化简单化。通常鼠标或热键的动作我们称之为事件（Event），而由鼠标或热键引发的一连串程序的动作，称之为事件驱动（Event Driver）。而对事件进行处理程序或函数，我们称之为事件处理程序（Event Handler）。

(3) 事件处理程序

在 JavaScript 中对象事件的处理通常由函数(Function)担任。其基本格式与函数全部一样，可以将前面所介绍的所有函数作为事件处理程序。格式如下：

Function 事件处理名（参数表）{

事件处理语句集;

……

}

(4) 事件驱动

JavaScript 事件驱动中的事件是通过鼠标或热键的动作引发的。它主要有以下几个事件：① 单击事件 onClick

当用户单击鼠标按钮时，产生 onClick 事件。同时 onClick 指定的事件处理程序或代码将被调用执行。通常在下列基本对象中产生：

button（按钮对象）

checkbox（复选框）或（检查列表框）

radio （单选钮）

reset buttons（重要按钮）

submit buttons（提交按钮）

例：可通过下列按钮激活 change()文件：

```
<Form>
<Input type="button" Value=“ ” onClick="change()">
</Form>
```

在 onClick 等号后，可以使用自己编写的函数作为事件处理程序，也可以使用 JavaScript 中内部的函数。还可以直接使用 JavaScript 的代码等。例：

<Input type="button" value=" " onclick=alert("这是一个例子")；

② onChange 改变事件

当利用 text 或 texturea 元素输入字符值改变时发该事件，同时当在 select 表格项中一个选项状态改变后也会引发该事件。例：

```
<Form>
```

```
<Input type="text" name="Test" value="Test" onCharge="check('this.test)">
</Form>
```

③ 选中事件 onSelect

当 Text 或 Textarea 对象中的文字被加亮后，引发该事件。

④ 获得焦点事件 onFocus

当用户单击 Text 或 textarea 以及 select 对象时，产生该事件。此时该对象成为前台对象。

⑤ 失去焦点 onBlur

当 text 对象或 textarea 对象以及 select 对象不再拥有焦点、而退到后台时，引发该文件，他与 onFocas 事件是一个对应的关系。

⑥ 载入文件 onLoad

当文档载入时，产生该事件。onLoad 一个作用就是在首次载入一个文档时检测 cookie 的值，并用一个变量为其赋值，使它可以被源代码使用。

⑦ 卸载文件 onUnload

当 Web 页面退出时引发 onUnload 事件，并可更新 Cookie 的状态。

4.范例

范例 1：下例程序是一个自动装载和自动卸载的例子。即当装入 HTML 文档时调用 loadform()函数,而退出该文档进入另一 HTML 文档时则首先调用 unloadform()函数,确认后方可进入。

```
test3_1.htm
<HTML>
<HEAD>
<script Language="JavaScript">
<!--
function loadform(){
alert("这是一个自动装载例子!");
}
function unloadform(){
alert("这是一个卸载例子!");
}
//-->
</Script>
</HEAD>
<BODY OnLoad="loadform()" OnUnload="unloadform()">
<a href="test.htm">调用</a>
</BODY>
</HTML>
```

范例 2：这是一个获取浏览器版本号的程序。该程序首先显示一个波浪一提示信息。之后显示浏览器的版本号有关信息。

test3_2.htm

```
<html>
<head>
<script language="JavaScript"><!--
// -->
```

```
function makeArray(n){
this.length=n
return this
}
function hexfromdec(num) {
hex=new makeArray(1);
var hexstring="";
var shifthex=16;
var temp1=num;
for(x=1; x>=0; x--) {
hex[x]=Math.round(temp1/shifthex - .5);
hex[x-1]=temp1 - hex[x] * shifthex;
temp1=hex[x-1];
shifthex /= 16;
}
for (x=1; x>=0; x--) { hexstring+=getletter(hex[x]); }
return (hexstring);
}

function getletter(num) {
if (num < 10) { return num; }
else {
if (num == 10) { return "A" }
if (num == 11) { return "B" }
if (num == 12) { return "C" }
if (num == 13) { return "D" }
if (num == 14) { return "E" }
if (num == 15) { return "F" }
}
}
function rainbow(text){
var color_d1;
var allstring="";
for(i=0;i<text.length;i=i+2){
color_d1=255*Math.sin(i/(text.length/3));
color_h1=hexfromdec(color_d1);
allstring+="<FONT COLOR="+color_h1+"ff"+color_h1+">"+text.substring(i,i+2)+"</FONT>";
}
return allstring;
}

function sizefont(text){
var color_d1;
var allstring="";
var flag=0;
for(i=0,j=0;i<text.length;i=i+1){
```

```
if (flag==0) {
j++;
if (j>=7) {
flag=1;}}
if (flag==1) {
j=j-1;
if (j<=0) {
flag=0; }}
allstring+="<FONT SIZE="+ j + ">" + text.substring(i,i+1) + "</FONT>";
}
return allstring;
}
document.write("<font size=8><CENTER>")
document.write("<BR><BR>")
document.write( sizefont("这是一个获取 WEB 浏览器的程序"))
document.write("</CENTER></font>")
document.write("浏览器名称: "+navigator.appName+"<br>");
document.write("版本号: "+navigator.appVersion+"<br>");
document.write("代码名字: "+navigator.appCodeName+"<br>");
document.write("用户代理标识: "+navigator.userAgent);
</script>
<body>
</body>
</html>
```

输出结果图 1 所示。

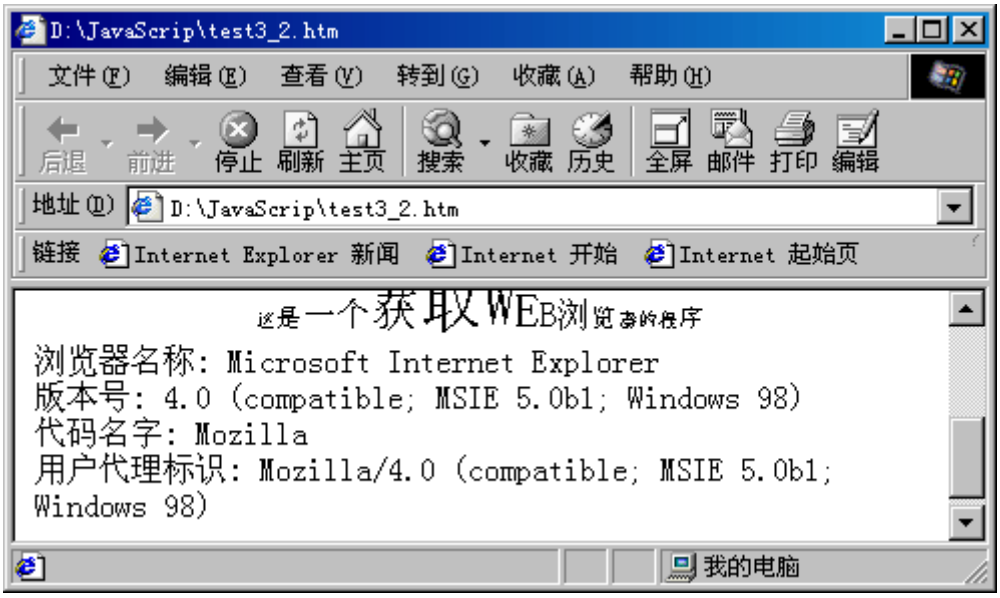


图 1

本讲介绍了 JavaScript 程序设计的有关内容。程序流、函数、事件是我们学习掌握 JavaScript 编程的重点。

基于对象的 JavaScript 语言

JavaScript 语言是基于对象的（Object-Based），而不是面向对象的（object-oriented）。之所以说它是一门基于对象的语言，主要是因为它没有提供象抽象、继承、重载等有关面向对象语言的许多功能。而是把其它语言所创建的复杂对象统一起来，从而形成一个非常强大的对象系统。　　虽然 JavaScript 语言是一门基于对象的，但它还是具有一些面向对象的基本特征。它可以根据需要创建自己的对象，从而进一步扩大 JavaScript 的应用范围，增强编写功能强大的 Web 文档。

一、对象的基础知识

1、对象的基本结构

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

2、引用对象的途径

一个对象要真正地被使用，可采用以下几种方式获得：

- o 引用 JavaScript 内部对象；
- o 由浏览器环境中提供；
- o 创建新对象。

这就是说一个对象在被引用之前，这个对象必须存在，否则引用将毫无意义，而出现错误信息。从上面中我们可以看出 JavaScript 引用对象可通过三种方式获取。要么创建新的对象，要么利用现存的对象。

3、有关对象操作语句

JavaScript 不是一纯面向对象的语言，它设有提供面向对象语言的许多功能，因此 JavaScript 设计者之所以把它你“基于对象”而不是面向对象的语言，在 JavaScript 中提供了几个用于操作对象的语句和关键字及运算符。

（1） For...in 语句

格式如下：

For（对象属性名 in 已知对象名）

说明：

o 该语句的功能是用于对已知对象的所有属性进行操作的控制循环。它是将一个已知对象的所有属性反复置给一个变量；而不是使用计数器来实现的。

o 该语句的优点就是无需知道对象中属性的个数即可进行操作。

例：下列函数是显示数组中的内容：

```
Function showData(object)
for (var X=0; X<30;X++)
document.write(object[i]);
```

该函数是通过数组下标顺序值，来访问每个对象的属性，使用这种方式首先必须知道数组的下标值，否则若超出范围，则就会发生错误。而使 For...in 语句，则根本不需要知道对象属性的个数，见下：

```
Function showData(object)
for(var prop in object)
document.write(object[prop]);
```

使用该函数时，在循环体中，For 自动将的属性取出来，直到最后为此。

（2） with 语句

使用该语句的意思是：在该语句体内，任何对变量的引用被认为是这个对象的属性，以节省一些代码。

with object{
...}
所有在 with 语句后的花括号中的语句，都是在后面 object 对象的作用域的。
（3） this 关键字
this 是对当前的引用，在 JavaScript 由于对象的引用是多层次，多方位的，往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，最后自己已不知道现在引用的那一个对象，为此 JavaScript 提供了一个用于将对象指定当前对象的语句 this。
（4） New 运算符
虽然在 JavaScript 中对象的功能已经是非常强大的了。但更强大的是设计人员可以按照需求来创建自己的对象，以满足某一特定的要求。使用 New 运算符可以创建一个新的对象。其创建对象使用如下格式：
Newobject=NEW Object(Parameters table);
其中 Newobject 创建的新对象：object 是已经存在的对象； parameters table 参数表；new 是 JavaScript 中的命令语句。
如创建一个日期新对象
newData=New Data()
birthday=New Data (December 12.1998)
之后就可使 NewData、birthday 作为一个新的日期对象了。

4、对象属性的引用

对象属性的引用可由下列三种方式之一实现：
（1）使用点（.）运算符

```
university.Name=“云南省”
university.city=“昆明市”
university.Date=″1999″
```

其中 university 是一个已经存在的对象，Name、City、Date 是它的三个属性，并通过操作对其赋值。
（2）通过对象的下标实现引用

```
university[0]=“云南”
university[1]=“昆明市”
university[2]=″1999″
```

通过数组形式的访问属性，可以使用循环操作获取其值。

```
function showunievsity(object)
for (var j=0;j<2; j++)
document.write(object[j])
```

若采用 For...in 则可以不知其属性的个数后就可以实现：

```
Function showmy(object)
for (var prop in this)
docament.write(this[prop]);
```

（3）通过字符串的形式实现

```
university[″Name″]=“云南”
```

```
university["City"]=“昆明市”
university["Date"]="1999"
```

5、对象的方法的引用

在 JavaScript 中对象方法的引用是非常简单的。

ObjectName.methods()

实际上 methods()=FunctionName 方法实质上是一个函数。 如引用 university 对象中的 showmy（）方法，则可使用：

document.write (university.showmy())

或：document.write(university)

如引用 math 内部对象中 cos()的方法

则：

```
with(math)
document.write(cos(35));
document.write(cos(80));
```

若不使用 with 则引用时相对要复杂些：

```
document.write(Math.cos(35))
document.write(math.sin(80))
```

二、常用对象的属性和方法

JavaScript 为我们提供了一些非常有用的常用内部对象和方法。用户不需要用脚本来实现这些功能。这正是基于对象编程的真正目的。

在 JavaScript 提供了 string（字符串）、math（数值计算）和 Date（日期）三种对象和其它一些相关的方法。从而为编程人员快速开发强大的脚本程序提供了非常有利的条件。

1、常用内部对象

在 JavaScript 中对于对象属性与方法的引用，有两种情况：其一是说该对象是静态对象，即在引用该对象的属性或方法时不需要为它创建实例；而另一种对象则在引用它的对象或方法是必须为它创建一个实例，即该对象是动态对象。

对 JavaScript 内部对象的引用，以是紧紧围绕着它的属性与方法进行的。因而明确对象的静动性对于掌握和理解 JavaScript 内部对象是具有非常重要的意义。

1）、串对象

- string 对象：内部静态性。
- 访问 properties 和 methods 时，可使用（.）运算符实现。
- 基本使用格式：objectName.prop/methods

（1）串对象的属性

该对象只有一个属性，即 length。它表明了字符串中的字符个数，包括所有符号。

例：

```
mytest="This is a JavaScript"
mystringlength=mytest.length
```

最后 mystringlength 返回 mytest 字串的长度为 20。

（2）串对象的方法

string 对象的方法共有 1 9 个。主要用于有关字符串在 Web 页面中的显示、字体大小、字体颜色、字符的搜索

以及字符的大小写转换。
其主要方法如下：

o 锚点 anchor()：该方法创建如用 Html 文档中一样的 anchor 标记。使用 anchor 如用 Html 中(A Name="")一样。通过下列格式访问:string.anchor(anchorName)。

o 有关字符显示的控制方法

big 字体显示, Italics()斜体字显示，bold()粗体字显示，blink()字符闪烁显示，small（）字符用小体字显示，fixed()固定高亮字显示、fontsize(size)控制字体大小等。

o 字体颜色方法；fontcolor(color)

o 字符串大小写转换

toLowerCase()－小写转换，toUpperCase()大写转换。下列把一个给定的串分别转换成大写和小写格式:
string=stringValue.toUpperCase 和 string=stringValue.toLowerCase。

o 字符搜索：indexOf[charactor,fromIndex]

从指定 formIndtx 位置开始搜索 charactor 第一次出现的位置。
返回字串的一部分字串：substring(start,end)
从 start 开始到 end 的字符全部返回。

2)、算术函数的 math 对象

功能：提供除加、减、乘、除以外的一引些自述运算。如对数，平方根等 。
静动性：静态对象
（1）主要属性

math 中提供了 6 个属性，它们是数学中经常用到的常数 E、以 10 为底的自然对数 LN10、以 2 为底的自然对数 LN2、3.14159 的 PI、1/2 的平方根 SQRT1-2,2 的平方根为 SQRT2。

（2）主要方法

绝对值：abs()
正弦余弦值：sin(),cos()
反正弦反余弦 :asin(), acos()
正切反正切：tan(),atan()
四舍五入：round()
平方根：sqrt()
基于几方次的值：Pow(base,exponent)
...

3)、日期及时间对象

功能：提供一个有关日期和时间的对象。
静动性：动态性，即必须使用 New 运算符创建一个实例。例:
MyDate=New Date()
Date 对象没有提供直接访问的属性。只具有获取和设置日期和时间的方法。
日期起始值:1770年1月1日00:00:00。

1. 获取日期的时间方法

getYear(): 返回年数
getMonth():返回当月号数
getDate(): 返回当日号数
getDay():返回星期几
getHours():返回小时数
getMintes(:返回分钟数
getSeconds():返回秒数

getTime() ： 返回毫秒数

（2）设置日期和时间：

setYear();设置年

setDate():设置当月号数

setMonth():设置当月份数

setHours():设置小时数

setMintes():设置分钟数

setSeconds():设置秒数

setTime ():设置毫秒数

...

2、JavaScript 中的系统函数

JavaScript 中的系统函数又称内部方法。它提供了与任何对象无关的系统函数，使用这些函数不需创建任何实例,可直接用。

1. 返回字符串表达式中的值：

方法名：eval（字串表达式），例：

```
test=eval("8+9+5/2");
```

2. 返回字符串 ASCI 码：

方法名：unEscape (string)

3. 返回字符的编码：

方法名：escape(character)

4. 返回实数：

```
parseFloat(floustring);
```

5. 返回不同进制的数：

```
parseInt(numbestring ,rad.X)
```

其中 radix 是数的进制，numbs 字符串数

三、范例

下面是一个时钟显示的 JavaScript 文档。在文档中用了非常多的函数。

Test4_1.htm

```
<html>
<head>
<style TYPE="text/css">
<style>
</style>
<title>时钟</title>
<script LANGUAGE="JavaScript">
function showClock() {
}
function hideClock() {
}
var timerID = null
var timerRunning = false
function stopClock() {
if(timerRunning)
```

```
clearTimeout(timerID);
timerRunning = false
document.clock.face.value = "";
}
function showTime() {
var now = new Date();
var year = now.getYear();
var month = now.getMonth() + 1;
var date = now.getDate();
var hours = now.getHours();
var mins = now.getMinutes();
var secs = now.getSeconds();
var timeVal = "";
timeVal += ((hours <= 12) ? hours : hours - 12);
timeVal += ((mins < 10) ? ":0" : ":") + mins;
timeVal += ((secs <= 10) ? ":0" : ":") + secs;
timeVal += ((hours < 12) ? "AM" : "PM");
timeVal += ((month < 10) ? " on 0" : " on ") + month + "-";
timeVal += date + "-" + year;
document.clock.face.value = timeVal;
timerID = setTimeout("showTime()", 1000);
timerRunning = true
}
function startClock() {
stopClock();
showTime();
}

function windowOpener( indexnum ){
var loadpos="date.html"+"#"+indexnum;
controlWindow=window.open(loadpos,"date","toolbar=no,location=no,
directories=no,status=no,menubar=no,scrollbars=yes,resizable=yes,
width=620,height=400");
}
</script>
</head>
<body onLoad="startClock()" >
<p align="center"><big><span style="background-color: rgb(45,45,45)">
<font face="Arial">form</font>   <font face="宋体">时钟</font>
</span></big></p>
<p align="center">  </p>
<div align="center"><center>
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><form NAME="clock" onSubmit="0">
<div align="center"><center><p><input TYPE="text" NAME="face" size="20"
```

```
VALUE style="background-color: rgb(192,192,192)"> </p>
</center></div>
</form>
</td>
</tr>
</table>
</center></div>
</body>
</html>
```

本讲介绍了基于对象的 JavaScript 中常用内部对象属性、方法的使用。

在 JavaScript 中创建新对象

使用 JavaScript 可以创建自己的对象。虽然 JavaScript 内部和浏览器本身的功能已十分强大，但 JavaScript 还是提供了创建一个新对象的方法。使其不必像超文本标识语言那样，求于或其它多媒体工具，就能完成许多复杂的工作。

在 JavaScript 中创建一个新的对象是十分简单的。首先它必须定义一个对象，而后再为该对象创建一个实例。这个实例就是一个新对象，它具有对象定义中的基本特征。

一、对象的定义

JavaScript 对象的定义，其基本格式如下：

```
Function Object（属性表）
This.prop1=prop1
This.prop2=prop2
...
This.meth=FunctionName1;
This.meth=FunctionName2;
...
```

在一个对象的定义中，可以为该对象指明其属性和方法。通过属性和方法构成了一个对象的实例。如以下是一个关于 University 对象的定义：

```
Function university(name,city,creatDate URL)
This.name=name
This.city=city
This.creatDate=New Date(creatDate)
This.URL=URL
```

其基本含义如下：

Name－指定一个“单位”名称。

City－“单位”所在城市。

CreatDate－记载 university 对象的更新日期。

URL－该对象指向一个网址。

二、创建对象实例

一旦对象定义完成后，就可以为该对象创建一个实例了：

```
NewObject=New object();
```

其中 Newobjet 是新的对象，Object 已经定义好的对象。例：

```
U1=New university(“云南省”，“昆明市”，"January 05,199712:00:00","http://www.YN.KM")
U2=New university(“云南电子科技大学”，“昆明”,"January 07,1997 12:00:00","htlp://www.YNKJ.CN")
```

三、对象方法的使用

在对象中除了使用属性外，有时还需要使用方法。在对象的定义中，我们看到 This.meth=FunctionName 语句，那就是为定义对象的方法。实质对象的方法就是一个函数 FunctionName，通过它实现自己的意图。

例在 university 对象中增加一个方法，该方法是显示它自己本身，并返回相应的字串。

```
function university(name,city,createDate,URL)
This.Name=Name;
This.city=city;
This.createDate=New Date(creatDate);
This.URL=URL;
This.showuniversity=showuniversity;
```

其中 This.showuniversity 就是定义了一个方法－－－showuniversity()。

而 showuniversity()方法是实现 university 对象本身的显示。

```
function showuniversity()
For (var prop in this)
alert(prop+="+this[prop]+"");
```

其中 alert 是 JavaScript 中的内部函数，显示其字符串。

四、JavaScript 中的数组

使用 New 创建数组

JavaScript 中没有提供像其它语言具有明显的数组类型，但可以通过 function 定义一个数组，并使用 New 对象操作符创建一个具有下标的数组。从而可以实现任何数据类型的存储。

a、定义对象的数组

```
Function arrayName(size){
This.length=Size;
for(var X=; X<=size;X++)
this[X]=0;
Reture this;
}
```

其中 arrayName 是定义数组的一个名子，Size 是有关数组大小的值（1-size），即数组元素的个数。通过 for 循环对一个当前对象的数组进行定义，最后返回这个数组。

从中可以看出，JavaScript 中的数组是从 1 到 size，这与其它 0 到 size 的数组表示方法有所不同，当然你可根据需要将数组的下标由 1 到 size 调整到 0 到 size－ 1 ,可由下列实现:

```
Function arrayName (size)
For (var X=0; X<=size;X++)
this[X]=0;
this.lenght=size;
Return this;
```

从上面可以看出该方法是只是调整了 this.lenght 的位置，该位置是用于存储数组的大小的。从而调整后的数组的下标将与其它语言一致。但请读者注意正是由于数组下标顺序由 1 到 size，使得 JavaScript 中的对象功能更加强大。

b、创建数组实例

一个数组定义完成以后，还不能马上使用，必须为该数组创建一个数组实例：
Myarray=New arrayName(n);
并赋于初值：
Myarray[1]=“字串 1 ”；
Myarray[2]=“字串 2 ”；
Myarray[3]=“字串 3 ”；
...
Myarray[n]=“字串 n”；
一旦给数组赋于了初值后，数组中就具有真正意义的数据了，以后就可以在程序设计过程中直接引用。
创建多维数组
```
Function creatMArray(row,col){
var indx=0;
this.length=(row*10)+col
for(var x=1;x<=row;x++)
for(var y=1;y<=col;y++)
indx=(x*10)+y;
this[indx]=” ” ;
}
myMArray=new creatMArray();
```
之后可通过 myMArray[11]、myMArray[12]、myMArray[13]、myMArray[21]、myMArray[22]、myMArray[23]、…来引用。
内部数组
在 Java 中为了方便内部对象的操作,可以使用窗体(Forms)、框架(Frames)、元素(element)、链接(links)和锚(Anchors)数组实现对象的访问。
anchors[]:使用《A name=“anchorName “》标识来建立锚的链接。

links[]: 使用<A href=”URL”>来定义一个越文本链接项。

Forms[]: 在程序中使用多窗体时,建立该数组。

Elements[]:在一个窗口中使用从个元素时,建立该数组。
Frames[]:建立框架时,使用该数组
anchors[]用于窗体的访问(它是通过《form name=“form1”》所指定的),link[]用于被链接到的锚点的访问(它是通过《a href=URL》所指定的)。Forms[]反映窗体的属性,而 anchors[]反映 Web 页面中的链接属性。
有关锚数组的文档：
```
<HTML>
<HEAD>
<BODY>
<A NAME=”MyAnchorsName1”> 定义第一个锚名
HTML Code
<A NAME=”MyAnchorsName2”> 定义第二个锚名
HTML Code
<A HREF=”#MyAnchorsName1”>建立锚的链接
<A HREF=”#MyAnchorsName2?gt; 建立锚的链接
….
```

该文档段建立了两面全锚的链接,可通过 Anchors[]访问这些锚。document.Anchors[0]反映第一个锚,而 document.Anchors[1]反映第二个锚的有关信息。

五、范例

范例 1：一个动态文字滚动的例子。

test5_1.htm

```
<html>
<head>
<title></title>
<script LANGUAGE="JavaScript">

with (top.window.location)
{baseURL = href.substring (0,href.lastIndexOf ("/") + 1)}
total_toc_items = 0;
current_overID = "";
last_overID = "";
browser = navigator.appName;
version = parseInt(navigator.appVersion);
client=null;
loaded = 0;
if (browser == "Netscape" && version >= 3) client = "ns3";
function toc_item (img_name,icon_col,width,height) {
if (client =="ns3") {
img_prefix = baseURL + img_name;
this.icon_col = icon_col;
this.toc_img_off = new Image (width,height);
this.toc_img_off.src = img_prefix + "_off.gif";
this.toc_img_on = new Image (width,height);
this.toc_img_on.src = img_prefix + "_on.gif";
}
}

function new_toc_item (img_name,icon_row,width,height) {
toc_item [img_name] = new toc_item (img_name,icon_row,width,height);
}

function toc_mouseover (itemID) {
if (client =="ns3") {
current_overID = itemID;
if (current_overID != last_overID) {
document [current_overID].src = toc_item [current_overID].toc_img_on.src;
if (last_overID != "") {
document.images [last_overID].src = toc_item[last_overID].toc_img_off.src;
}
last_overID = current_overID;
}
}
```

```
}

function toc_mouseout () {
if (client =="ns3") {
if (current_overID != "") {
document.images [current_overID].src = toc_item [current_overID].toc_img_off.src;
}
current_overID = "";
last_overID = "";
}
}
new_toc_item ("1",2,120,20);
<!-- Begin
function bannerObject(p){
this.msg = MESSAGE
this.out = " "
this.pos = POSITION
this.delay = DELAY
this.i = 0
this.reset = clearMessage}

function clearMessage(){
this.pos = POSITION}
var POSITION = 50;
var DELAY = 150;
var MESSAGE = "这是一个动态 JavaScript 文字显示的例子";
var scroll = new bannerObject();
function scroller(){
scroll.out += " ";
if(scroll.pos>0)
for (scroll.i = 0; scroll.i < scroll.pos; scroll.i++) { scroll.out +=" " ; }
if (scroll.pos>= 0)
scroll.out += scroll.msg
else
scroll.out = scroll.msg.substring(-scroll.pos,scroll.msg.length)
document.noticeForm.notice.value = scroll.out
scroll.out = " ";
scroll.pos--;
scroll.pos--;
if (scroll.pos < -(scroll.msg.length)) { scroll.reset(); } setTimeout
('scroller()',scroll.delay);}
</script>
</head>
<body onload="scroller()" bgcolor="#000000" link="#C0C0C0" vlink="#C0C0C0"
alink="#008080"
text="#C0C0C0">
```

```
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><form NAME="noticeForm">
<p><input TYPE="text" name="notice" size="40" style="background-color: rgb(192,192,192)"></p>
</form>
</td>
</tr>
</table>
</center></div>
</body>
</html>
```

范例 2:颜色变化的例子。

test5_2.htm

```
<html>
<head>
<script>
<!--
function makearray(n) {
this.length = n;
for(var i = 1; i <= n; i++)
this[i] = 0;
return this;}
hexa = new makearray(16);
for(var i = 0; i < 10; i++)
hexa[i] = i;
hexa[10]="a";
hexa[11]="b";
hexa[12]="c";
hexa[13]="d";
hexa[14]="e";
hexa[15]="f";
function hex(i) {
if (i < 0)
return "00";
else if (i > 255)
return "ff";
else return "" + hexa[Math.floor(i/16)] + hexa[i%16];}
function setbgColor(r, g, b) {
var hr = hex(r);
var hg = hex(g);
var hb = hex(b);
document.bgColor = "#"+hr+hg+hb;}
function fade(sr, sg, sb, er, eg, eb, step) {
for(var i = 0; i <= step; i++) {
setbgColor( Math.floor(sr * ((step-i)/step) + er * (i/step)),
```

```
Math.floor(sg * ((step-i)/step) + eg * (i/step)), Math.floor(sb *
((step-i)/step) + eb * (i/step))); }}
function fadein() {
fade(255,0,0,0,0,255,100);
fade(0,0,255,0,255,0,100);
fade(0,255,0, 0,0,0, 100);}
fadein();
// -->
</script>
<body>
</body>
</html>
```

本讲介绍了用户自行创建对象的方法，用户可根据需要创建自己的对象。并介绍了 JavaScript 中建数组的方法。

JavaScript 对象系统的使用

使用浏览器的内部对象系统，可实现与 HTML 文档进行交互。它的作用是将相关元素组织包装起来，提供给程序设计人员使用，从而减轻编程人的劳动，提高设计 Web 页面的能力。

一、浏览器对象层次及其主要作用

除了前面提到过的文档 document 对象外，Navigator 浏览器中还提供了窗口(Window)对象以及历史(History)和位置（Location）对象。

浏览器对象(Navigator)

提供有关浏览器的信息

窗口对象(Windows)

Window 对象处于对象层次的最顶端，它提供了处理 Navigator 窗口的方法和属性。

位置对象(Location)

Location 对象提供了与当前打开的 URL 一起工作的方法和属性，它是一个静态的对象。

历史对象(History)

History 对象提供了与历史清单有关的信息。

文档对象(Document)

document 对象包含了与文档元素(elements)一起工作的对象，它将这些元素封装起来供编程人员使用。

编程人员利用这些对象，可以对 WWW 浏览器环境中的事件进行控制并作出处理。在 JavaScript 中提供了非常丰富的内部方法和属性,从而减轻了编程人员的工作,提高编程效率。这正是基于对象与面向对象的根本区别所在。在这些对象系统中,文档对象属于非常重要的,它位于最低层,但对于我们实现 Web 页面信息交互起作关键作用。因而它是对象系统的核心部分。

二、文档对象功能及其作用

在 Navigator 浏览器中，document 文档对象是核心是，同时也是最重要的。见表 6-1 所示。

Links	Anchor	Form	Method	Prop
链接对象	锚对象	窗体对象	方法	对象

表 6－1 document 对象

从表 6-1 中可以看出，document 对象的主要作用就是把这些基本的元素（如 links,anchor 等）包装起来，提

供给编程人员使用。从另一个角度看，document 对象中又是由属性和方法组成。

1、document 中三个主要的对象

在 document 中主要有：links,anchor,form 等三个最重要的对象：

（1）anchor 锚对象：

anchor 对象指的是<A Name=...></A>标识在 HTML 源码中存在时产生的对象。它包含着文档中所有的 anchors 信息。

（2）链接 links 对象

link 对象指的是用<A Href=...> </A>标记的连接一个超文本或超媒体的元素作为一个特定的 URL。

（3）窗体（Form）对象

窗体对象是文档对象的一个元素，它含有多种格式的对象储存信息，使用它可以在 JavaScript 脚本中编写程序进行文字输入，并可以用来动态改变文档的行为。通过 document. Forms[]数组来使得在同一个页面上可以有多个相同的窗体,使用 forms[]数组要比使用窗体名字要方便得多。

例:下面就是一个使用窗体数组和窗体名字的例子。该程序使得两个窗体中的字段内容保持一致。

Test6_1.htm

```
<Html>
<head>
</head>
<body>
<form >
<input type=text onChange="document.my.elements[0].value=this.value;" >
</form>
<form NAME="my">
<input type=text onChange="document.forms[0].elements[0].value=this.value;">
</form>
</body>
</html>
```

其中用了 OnChnge 事件(当窗体内容改变时激发)。第一个使用窗体名字标识 my,第二个使用窗体数组 Forms[]。其效果是一致。

2、文档对象中的 attribute 属性

document 对象中的 attribute 属性，主要用于在引用 Href 标识时，控制着有关颜色的格式和有关文档标题、文档原文件的 URL 以及文档最后更新的日期。这部分元素的主要含义如下：

（1）链接颜色：alinkcolor

这个元素主要用于，当选取一个链接时，链接对象本身的颜色就按 alinkcolo r 指定改变。

（2）链接颜色：linkcolor

当用户使用<A Href=...> Text string </A>链接后，Textstring 的颜色就会按 Linkcolor 所指定的颜色更新。

（3）浏览过后的颜色：VlinkColor

该属性表示的是已被浏览存储为已浏览过的链接颜色。

（4）背景颜色：bgcolor

该元素包含文档背景的颜色。

（5）前景颜色：Fgcolor

该元素包含 HTML 文档中文本的前景颜色。

3、文档对象的基本元素

（1）窗体属性：

窗体属性是与 HTML 文档中<Form>...</Form>相对应的一组对象在 HTML 文档所创建的窗体数，由 length 指定。通过 document.forms.length 反映该文档中所创建的窗体数目。

（2）锚属性：anchors

该属性中，包含了 HTML 文档的所有<A> </A>标记为 Name=...的语句标识。所有“锚”的数目保存在 document.anchors.length 中。

（3）链接属性：links

链接属性是指在文档中<A>...</A>的由 Href=...指定的数目，其链接数目保存在 document.links.length 中。

三、范例

例 1:下面我们通过一个例子来说明文档对象的综合应用。输出结果见图 6-2 所示。

Test6_2.htm

```
<html>
<head>
</HEAD>
<BOdy>
<Form Name="mytable">
请输入数据：
<Input Type="text" Name="text1" Value="">
</Form>
<A name="Link1" href="test31.htm">链接到第一个文本</a><br>
<A name="Link2" href="test32.htm">链接到第二个文本</a><br>
<A name="Link2" href="test33.htm">链接到第三个文本</a><br>
<A href="#Link1">第一锚点</a>
<A href="#Link2">第二锚点</a>
<A Href="#Link3">第三锚点</a>
<BR>
<Script Language="JavaScript">
document.write("文档有"+document.links.length+"个链接"+"<br>");
document.write("文档有"+document.anchors.length+"个锚点"+"<br>");
document.write("文档有"+document.forms.length+"个窗体");
</script>
</body>
</HTML>
```

图 6-2

例子 2：下列程序随机产生每日一语。

test6_3.html

```
<HTML>
<HEAD>
<script Language="JavaScript">
<!--
tips = new Array(6);
tips[0]="每日一语（1）";
tips[1]="每日一语（2）";
tips[2]="每日一语（3）";
tips[3]="每日一语（4）";
tips[4]="每日一语（5）";
tips[5]="每日一语（6）";
index = Math.floor(Math.random() * tips.length);
document.write("<FONT SIZE=8 COLOR=DARKBLUE>" + tips[index]+"</FONT>");
</Script>
</HEAD>
</BODY>
</HTML>
```

输出结果见图 6-3 所示。

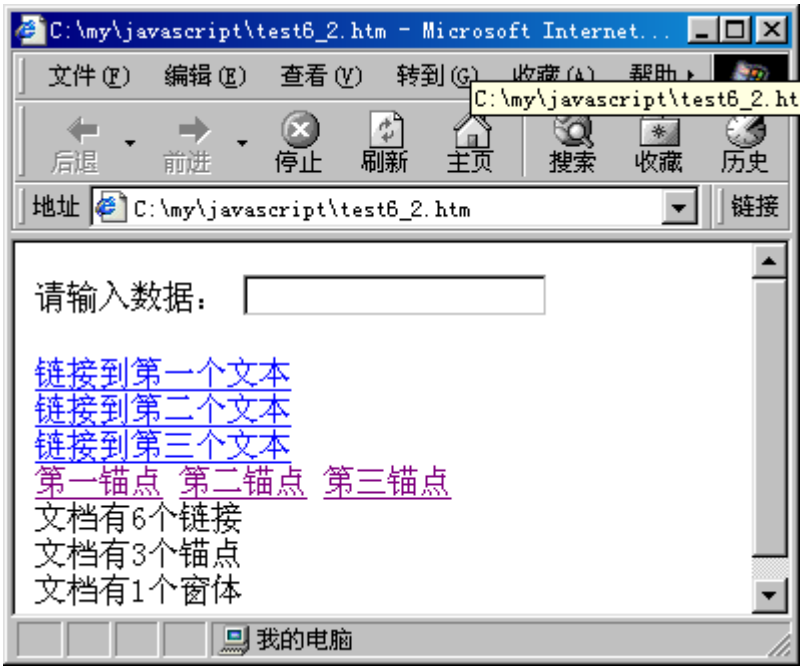


图 6-3

本讲主要介绍了 JavaScript 对象系统的使用方法，其中重点介绍了文档对象及使用。

JavaScript 窗口及输入输出

JavaScript 是基于对象的脚本编程语言，那么它的输入输出就是通过对象来完成的。其中有关输入可通过窗口（Window）对象来完成，而输出可通过文档（document）对象的方法来实现。

一、窗口及输入输出

请看下面例子：

```
<HTML>
<Head>
<script languaga="JavaScript">
Var test=window.prompt("请输入数据:");
document.write(test+"JavaScript 输入输出的例子");
</script>
</Head>
</HTML>
```

其中 window.prompt()就是一个窗口对象的方法，其基本作用是，当装入 Web 页面时在屏幕上显示一个具有“确定”和“取消”的对话框，让你输出数据。document.writle 是一个文档对象的方法，它的基本功能，是实现 Web 页面的输出显示。见图 1 所示。

图 1

1、窗口对象

该对象包括许多有用的属性、方法和事件驱动程序，编程人员可以利用这些对象控制浏览器窗口显示的各个方面，如对话框、框架等。在使用应注意以下几点：

该对象对应于 HTML 文档中的<Body>和<FrameSet>两种标识；

onload 和 onunload 都是窗口对象属性；

在 JavaScript 脚本中可直接引用窗口对象。如：

window.alert("窗口对象输入方法")

可直接使用以下格式：

alert("窗口对象输入方法")

2、窗口对象的事件驱动

窗口对象主要有装入 Web 文档事件 onload 和卸载时 onunload 事件。用于文档载入和停止载入时开始和停止更新文档。

3、窗口对象的方法

窗口对象的方法主要用来提供信息或输入数据以及创建一个新的窗口。

创建一个新窗口 open()

使用 window.open（参数表）方法可以创建一个新的窗口。其中参数表提供有窗口的主要特性和文档及窗口的命名。

具有 OK 按钮的对话框

alert()方法能创建一个具有 OK 按钮的对话框。

具有 OK 和 Cancel 按钮的对话框

confirm()方法为编程人员提供一个具有两个按钮的对话框。

具有输入信息的对话框

prompt()方法允许用户在对话框中输入信息，并可使用默认值，其基本格式如下 prompt（“提示信息”，默认值）。

4、窗口对象中的属性

窗口对象中的属性主要用来对浏览器中存在的各种窗口和框架的引用，其主要属性有以下几个：

（1）frames 确文档中帧的数目

frames（帧）作为实现一个窗口的分隔操作，起到非常有用的作用，在使用注意以下几点：

frames 属性是通过 HTML 标识<Frames>的顺序来引用的，它包含了一个窗口中的全部帧数。

帧本身已是一类窗口，继承了窗口对象所有的全部属性和方法。

（2）Parent 指明当前窗口或帧的父窗口。

（3）defaultstatus:默认状态，它的值显示在窗口的状态栏中。

（4）status:包含文档窗口中帧中的当前信息。

（5）top:包括的是用以实现所有的下级窗口的窗口。

（6）window.指的是当前窗口

（7）self:引用当前窗口。

5、输出流及文档对象

在 JavaScript 文档对象中，提供了用于显示关闭、消除、打开 HTML 页面的输出流。

（1）创建新文档 open()方法

使用 document.open()创建一个新的窗口或在指定的命令窗口内打开文档。由于窗口对象是所加载的父对象，因而我们在调用它的属性或方法时，不需要加入 Window 对象。例用 Window. Open()与 Open()是一样的。

打开一个窗口的基本格式：

Window .open("URL","窗口名字","窗口属性"]

window 属性参数是由一个字符串列表项它由逗号分隔，它指明了有关新创建窗口的属性。见表 7-1 所示。

表 7-1

参 数	设定值	含 义
toolbar	yes/no	建立或不建立标准工具条
location	yes/no	建立或不建立位置输入字段
directions	yes/no	建立或不建立标准目录按钮
status	yes/no	建立或不建立状态条
menubar	yes/no	建立或不建立菜单条
scrollbar	yes/no	建立或不建立滚动条
revisable	yes/no	能否改变窗口大小
width	yes/no	确定窗口的宽度
Height	yes/no	确定窗口的高度。

在使用 Open()方法时，需要注意以下点。

通常浏览器窗中，总有一个文档是打开的。因而不需要为输出建立一个新文档。

在完成对 Web 文档的写操作后，要使用或调用 close()方法来实现对输出流的关闭。

在使用 open()来打开一个新流时，可为文档指定一个有效的文档类型，有效文档类型包括 text/HTML、text/gif、text/xim、text/plugin 等。

（2）write()、writeln()输出显示。

该方法主要用来实现在 Web 页面上显示输出信息。在实际使用中，需注意以下几点：

writeln()与 write()唯一不同之处在于在末尾加了一个换符。

为了正常显示其输出信息，必须指明<pre> </Pre>标记，使之告诉编辑器。

输出的文档类型，可以由浏览器中的有效的合法文本类型所确定。

（3）关闭文档流 close()

在实现多个文档对象中，必须使用 close()来关闭一个对象后，才能打开另一个文档对象。

（4）清除文档内容 clear()

使用该方法可清除已经打开文档的内容。

二、简单的输入、输出例子

在 JavaScript 中可以非常方便地实现输入输出信息，并与用户进行交互。

1、JavaScript 信息的输入

通过使用 JavaScript 中所提供的窗口对象方法 prompt()，就能完成信息的输入。该方法提供了最简便的信息输入方式，其基本格式如下：

Window.prompt("提示信", 预定输入信息);

此方法首先在浏览器窗口中弹出一个对话框，让用户自行输入信息。一旦输入完成后，就返回用户所输入信息的值。例:

test=prompt（“请输入数据:”，”this is a JavaScript”）

实际上 prompt()是窗口对象的一个方法。因为缺省情况下所用的对象就是 window 对象，所以 windows 对象可以省略不写。

2、输出显示

每种语言，都必须提供信息数据的输出显示。JavaScript 也是一样，它提供有几个用于信息输出显示的方法。比较常用的有 window.alert()、document.write 和及 document.writln()方法。

1)、document.write()方法和 document.writeln()方法

document 是 JavaScript 中的一个对象在它中封装许多有用的方法，其中 write()和 writeln()就是用于将文本信息直接输出到浏览器窗口中的方法。

document.write();

document.writeln();

说明：

write()和 writeln()方法都是用于向浏览器窗口输出文本字串；

二者的唯一区别就是 writeln()方法自动在文本之后加入回车符。

2)、 window.alert()输出

在 JavaScript 为了方便信息输出，JavaScript 提供了具有独立的对话框信息输出—alert()方法。alert()方法是 window 对象的一个方法，因此在使用时，不需要写 window 窗口对象名，而是直接使用就行了。它主要用途用在输出时产生有关警告提示信息或提示用户，一旦用户按“确定”钮后，方可继续执行其他脚本程序。

例:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<Script Language ="JavaScript">
alert("这是一个 JavaScript 测试程序");
</Script>
</BODY>
</HTML>
```

3)、利用输入、输出方法实现交互

在 JavaScript 中，可以利用 prompt()方法和 write()方法实现与 Web 页面用户进行交互。例下面就是一个有关实现交互的例子。

Test7_1.htm

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<Script Language="JavaScript">
<!-- Hide From Other Browsers
document.write("<H1>有关交互的例子");
my=prompt("请输入数据:");
```

```
document.write(my+"</H1>");
document.close();
// Stop Hiding from Other Browsers-->
</Script>
</BODY>
</HTML>
```

从上面程序可以看出：

可通过 write()和 prompt()方法实现交互。

在 JavaScript 脚本语言中可以使用 HTML 标识语言的代码。从而实现混合编程。其中<H1>和
就是 HTML 标识符。

四、范例

下列程序演示了你进入主页所停留的时间。

test7_2.htm

```
<html>
<form name="myform">
<td vAlign="top" width="135">您在此停留了：
<input name="clock" size="8" value="在线时间"></td>
</form>
<script language="JavaScript">
var id, iM = 0, iS = 1;
start = new Date();
function go()
{
now = new Date();
time = (now.getTime() - start.getTime()) / 1000;
time = Math.floor( time);
iS = time % 60;
iM = Math.floor( time / 60);
if ( iS < 10)
document.myform.clock.value = " " + iM + " 分 0" + iS + " 秒";
else
document.myform.clock.value = " " + iM + " 分 " + iS + " 秒";
id = setTimeout( "go()", 1000);
}
go();
</script>
</body>
</html>
```

在浏览器中的结果，见图 2 所示。

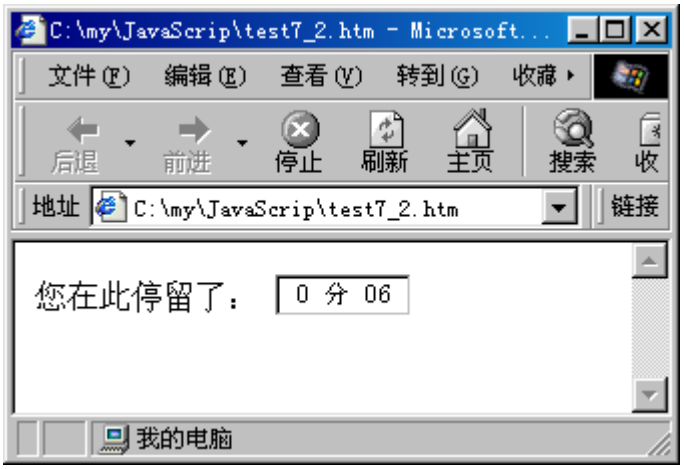


图 1

用 JavaScript 脚本实现 Web 页面信息交互

要实现动态交互，必须掌握有关窗体对象(Form)和框架对象(Frames)更为复杂的知识。

一、窗体基础知识

窗体对象可以使设计人员能用窗体中不同的元素与客户机用户相交互，而用不着在之前首先进行数据输入，就可以实现动态改变 Web 文档的行为。

1、什么是窗体对象

窗体（Form）:它构成了 Web 页面的基本元素。通常一个 Web 页面有一个窗体或几个窗体，使用 Forms[]数组来实现不同窗体的访问。

```
<form Name=Form1>
<INPUT type=text...>
<Input type=text...>
<Inpup byne=text...>
</form>
<form Name=Form2>
<INPUT type=text...>
<Input type=text...>
</form>
```

在 Forms[0]中共有三个基本元素，而 Forms[1]中只有两个元素。

窗体对象最主要的功能就是能够直接访问 HTML 文档中的窗体，它封装了相关的 HTML 代码：

```
<Form
Name ="表的名称"
Target ="指定信息的提交窗口"
action ="接收窗体程序对应的 URL"
Method =信息数据传送方式(get/post)
enctype ="窗体编码方式"
[onsubmit ="JavaScript 代码"]>
</Form>
```

2、窗体对象的方法

窗体对象的方法只有一个--submit()方法，该方法主要功用就是实现窗体信息的提交。如提交 Mytest 窗体,则使用下列格式：

document.mytest.submit()

3、窗体对象的属性

窗体对象中的属性主要包括以下：elements name action target encoding method.

除 Elements 外，其它几个均反映了窗体中标识中相应属性的状态，这通常是单个窗体标识；而 elements 常常是多个窗体元素值的数组，例：

elements[0].Mytable.elements[1]

4、访问窗体对象

在 JavaScript 中访问窗体对象可由两种方法实现：

（1）通过访问窗体

在窗体对象的属性中首先必须指定其窗体名，而后就可以通过下列标识访问窗体如：document.Mytable()。

（2）通过数组来访问窗体

除了使用窗体名来访问窗体外，还可以使用窗体对象数组来访问窗体对象。但需要注意一点，因窗体对象是由浏览器环境的提供的，而浏览器环境所提供的数组下标是由 0 到 n。所以可通过下列格式实现窗体对象的访问：

document.forms[0]

document.forms[1]

document.forms[2]...

5、引用窗体的先决条件

在 JavaScript 中要对窗体引用的条件是：必须先在页面中用标识创建窗体，并将定义窗体部分放在引用之前。

二、窗体中的基本元素

窗体中的基本元素由按钮、单选按钮、复选按钮、提交按钮、重置按钮、文本框等组成。

在 JavaScript 中要访问这些基本元素，必须通过对应特定的窗体元素的数组下标或窗体元素名来实现。每一个元素主要是通过该元素的属性或方法来引用。其引用的基本格式见下:

formName.elements[].methadName (窗体名.元素名或数组.方法)

formName.elemaent[].propertyName(窗体名.元素名或数组.属性)

下面分别介绍：

1、Text 单行单列输入元素

功能：对 Text 标识中的元素实施有效的控制。

基本属性：

Name：设定提交信息时的信息名称。对应于 HTML 文档中的 Name。

Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息。

defaultvalue：包括 Text 元素的默认值

基本方法：

blur()：将当前焦点移到后台。

select()：加亮文字。

主要事件：

onFocus：当 Text 获得焦点时，产生该事件。
OnBlur：从元素失去焦点时，产生该事件。
Onselect：当文字被加亮显示后，产生该文件。
onchange：当 Text 元素值改变时，产生该文件。
例：...

```
<Form name="test">
<input type="text" name="test" value="this is a javascript" >
</form>
...
<script language ="Javascirpt">
document.mytest.value="that is a Javascript";
document.mytest.select();
document.mytest.blur();
</script>
```

2、textarea 多行多列输入元素

功能：实施对 Textarea 中的元素进行控制。

基本属性

name：设定提交信息时的信息名称，对应 HTML 文档 Textarea 的 Name。
Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息。
Default value：元素的默认值。

方法：

blur()：将输入焦点失去
select()：将文字加亮后

事件：

onBlur:当失去输入焦点后产生该事件
onFocus:当输入获得焦点后，产生该文件
Onchange:当文字值改变时，产生该事件
Onselect:当文字加亮后，产生该文件

3、Select 选择元素

功能：实施对滚动选择元素的控制。

属性：

name:设定提交信息时的信息名称，对应文档 select 中的 name。
Length:对应文档 select 中的 length
options:组成多个选项的数组
selectIndex;该下标指明一个选项
select 在中每一选项都含有以下属性：
Text:选项对应的文字
selected:指明当前选项是否被选中
Index:指明当前选项的位置
defaultselected:默认选项

事件：

OnBlur：当 select 选项失去焦点时，产生该文件。
onFocas:当 select 获得焦点时，产生该文件。

Onchange:选项状态改变后，产生该事件。

4、Button 按钮

功能：实施对 Button 按钮的控制。

属性：

Name:设定提交信息时的信息名称，对应文档中 button 的 Name。

Value:用以设定出现在窗口中对应 HTML 文档中 Value 的信息。

方法：

click()该方法类似于一个按下的按钮。

事件：

onclick 当单击 button 按钮时，产生该事件。

例 ：

```
<Form name="test">
<input type="button" name="testcall" onclick=tmyest()>
</form>
...
<script language="javascirpt">
document.elements[0].value="mytest"; //通过元素访问
或
document.testcallvalue="mytest"; // 通过名字访问
</script>
.....
```

5、checkbox 检查框

功能：实施对一个具有复选框中元素的控制。

属性：

name:设定提交信息时的信息名称。

Value:用以设定出现在窗口中对应 HTML 文档中 Value 的信息。

Checked:该属性指明框的状态 true/false.

defauitchecked:默认状态

方法：

click()该方法使得框的某一个项被选中。

事件：

onclick:当框的选被选中时，产生该事件。

6、radio 无线按钮

功能：实施对一个具单选功能的无线按钮控制。

属性：

name:设定提交信息时的信息名称，对应 HTML 文档中的 radio 的 name 相同

value:用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中的 radio 的 name。

length:单选按钮中的按钮数目。

defalechecked:默认按钮。

checked:指明选中还是没有选中。

index:选中的按钮的位置。

方法：

chick():选定一个按钮。

事件：

onclick:单击按钮时，产生该事件。

7、hidden:隐藏

功能：实施对一个具有不显示文字并能输入字符的区域元素的控制。

属性：

name:设定提交信息时的信息名称，对应 HTML 文档的 hidden 中的 Name。

Value:用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档 hidden 中的 value。

defaleitvalue:默认值

8、Password 口令

功能：实施对具有口令输入的元素的控制。

属性：

Name:设定提交信息时的信息名称，对应 HTML 文档中 password 中的 name。

Value:用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中 password 中的 Value。

defaultvalu:默认值

方法

select():加亮输入口令域。

blur():使这丢失 passward 输入焦点。

focus():获得 password 输入焦点。

9、submit 提交元素

功能：实施对一个具有提交功能按钮的控制。

属性：

name:设定提交信息时的信息名称，对应 HTML 文档中 submit。

Value:用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中 value。

方法

click()相当于按下 submit 按钮。

事件：

onclick()当按下该按钮时，产生该事件。

三、范例

下面我们演示通过点击一个按钮（red）来改变窗口颜色，点击“调用动态按钮文档”调用一个动态按钮文档。

test8_1.htm

```
<html>
<head>
<Script Language="JavaScript">
//原来的颜色
document.bgColor="blue";
document.vlinkColor="white";
document.linkColor="yellow";
document.alinkcolor="red";
//动态改变颜色
function changecolor(){
```

```
document.bgColor="red";
document.vlinkColor="blue";
document.linkColor="green";
document.alinkcolor="blue";
}
</script>
</HEAD>
<body bgColor="White" >
<A href="test8_2.htm"> 调用动态按钮文档</a>
<form >
<Input type="button" Value="red" onClick="changecolor()">
</form>
</BODY>
</HTML>
```

输出结果见图 1 所示。

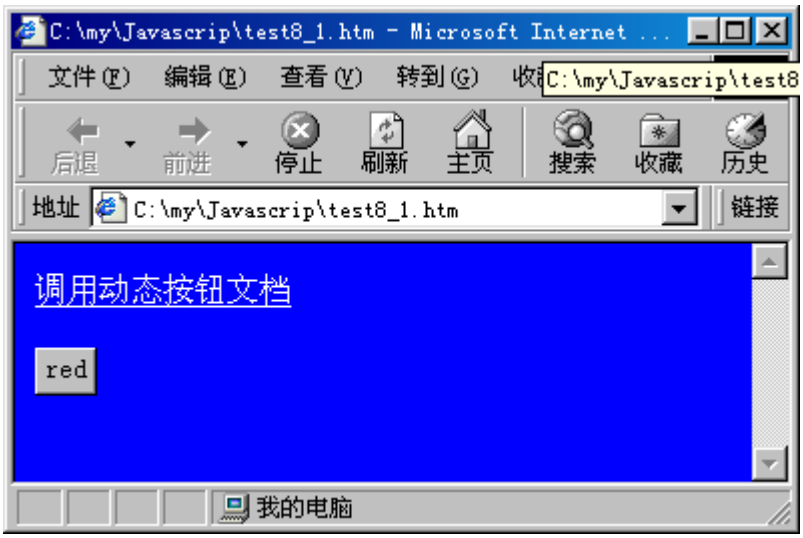


图 1

动态按钮程序。

test8_2.htm

```
<HTML>
<HEAD>
</HEAD>
<p align="center"> </p>
<div align="center"><center>
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><form name="form2" onSubmit="null">
<p><input type="submit" name="banner" VALUE="Submit"
onClick="alert('You have to put an \'action=[url]\' on the form
```

```
tag!!')"> <br>
<script language="JavaScript">
var id,pause=0,position=0;
function banner() {
// variables declaration
var i,k,msg=" 这里输入你要的内容";// increase msg
k=(30/msg.length)+1;
for(i=0;i<=k;i++) msg+=" "+msg;
// show it to the window
document.form2.banner.value=msg.substring(position,position-30);
// set new position
if(position++==msg.length) position=0;
// repeat at entered speed
id=setTimeout("banner()",60); }
// end -->
banner();
</script></p>
</form>
</td>
</tr>
</table>
</center></div>
<p>  </p>
<BODY>
<A href="test8_1.htm"> 返回</a>
</BODY>
</HTML>
```

输出结果见图 2 所示。

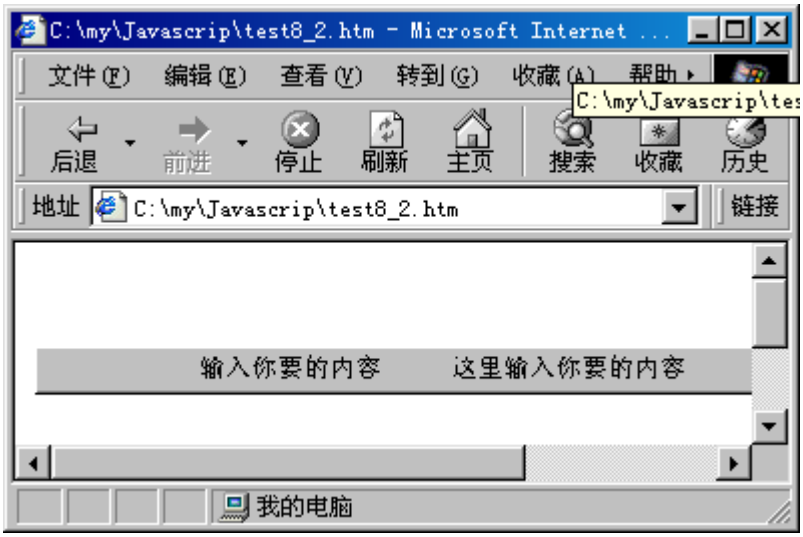


图 2

本讲介绍了使用 JavaScript 脚本实现 Web 页面信息交互的方法。其中主要介绍了窗体中的基本元素的主要功能和使用。

用 JavaScript 实现更复杂的交互

一、什么是框架

框架 Frames 最主要功用是"分割"视窗，使每个"小视窗"能显示不同的 HTM L 文件，不同框架之间可以互动(interact)，这就是说不同框架之间可以交换讯息与资料。例如:假设您开了两个 frames，第一个 frame 可显示书的目录，第二个 frame 则显示章节的具体内容。

框架可以将屏幕分割成不同的区域，每个区域有自己的 URL，通过 Frames[]数组对象来实现不同框架的访问。实际上框架对象本身也一类窗口，它继承了窗口对象的所有特征，并拥有所有的属性和方法。下面我们先看一下框架的例子。见图 9-1 所示。

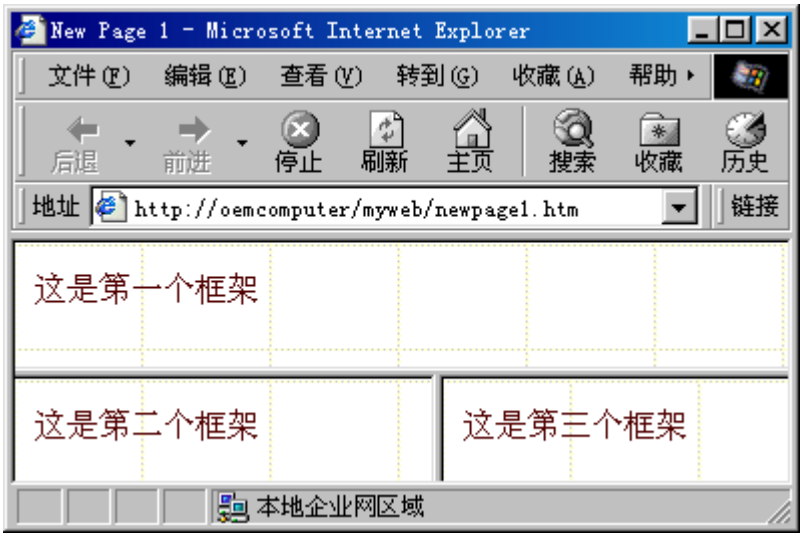


图 9-1 框架对象

```
<HTML>
<HEAD>
</HEAD>
<Frameset Rows="20%,80%">
<frame src="test9_1.html">
<Frameset Cols="50%,50%">
<frame src="test9_2.html">
<frame src="test9_3.html">
</Frameset>
</Frameset>
</HTML>
```

以上 HTML 标识将屏幕分成三个框架。先将窗口分成以二行为单位的窗口，之后再按分成二个窗口。并在相应的框架中放入自己的 HTML 文档。

通过[Framset]告诉浏览器您要设置几个框架；rows 这项参数告诉浏览器您想将视窗分割成几列；而 cols 这项参数是告诉浏览器您想将视窗分割成几行。

可以用很多组的 <frameset...> tags 将视窗分割得更复杂。

可以给每个 frame 一个"名字"(name)。frame 的名字在 JavaScript 语法中的地位非常重要。 可以用 <src> 告诉浏览器您要载入哪一个 HTML 文件。

二、如何访问框架

在前面我们介绍过使用 document.forms[]实现单一窗体中不同元素的访问。而要实现框架中多窗体的不同元素的访问，则必须使用 window 对象中的 Frames 属性。Frames 属性同样也是一个数组，他在　　父框架集中为每一个子框架设有一项。通过下标实现不同框架的访问:

parent.frames[Index1].docuement.forms[index2]

通过 parent.frames.length 确定窗口中窗体的数目。 除了使用数组下标来访问窗体外还可以使用框架名和窗体名来实现各元素的访：

parent.framesName.decument.formNames.elementName.(m/p)

三、范例

下面我们通过一个具体的实例, 来说明利用 JavaScript 脚本在 WEB 中实现更为复杂的信息交互。该例子是在一个多窗口中实现窗体信息的动态交互,在程序中首先在浏览器窗口中制作三个用于窗体交互的窗口,每个窗体窗口实现不同信息的动态交互。

tset9.html 为主调用文档它首先将窗口划分为具有二行的窗体,尔后再将第二行的窗体划分为具有二列的窗体;

test9-1.html 为显示标题文档;

test9_2.html 为第二框架文档其中需要注意的是:

通过 JavaScript 脚本将所示的“云南省”和“四川省”分别改为“昆明市”和“成都市”;

test7_3.html 为第三框架文档。

主调文档

主要作用是将窗口划分为具有二行的窗体,尔后再将第二行的窗体划分为具有二列的窗体。

Test9.htm

```
<HTML>
<HEAD>
</HEAD>
<Frameset Rows="10%,90%">
<frame src="test9_1.htm">
<Frameset Cols="40%,60%">
<frame src="test9_2.htm">
<frame src="test9_3.htm">
</Frameset>
</Frameset>
</HTML>
```

第一个框架

主要作用是显示标题文档。

Test9_1.htm

```
<HTML>
<HEAD>
</HEAD>
<H2>使用框架实现 WEB 交互</H2>
</HTML>
```

第二个框架

主要作用是实现交互。可以通过 JavaScript 脚本将所示的“云南省”和“四川省”分别改为“昆明市”和“成都

市”。

Test9_2.htm

```
<HTML>
<HEAD>
</HEAD>
<Body>
<Form name="test9_1">
请选择城市：<BR>
<Select name="select1" Multiple>
<Option>云南省
<Option>四川省
<Option>贵州省
<Option>山东省
<Option>江苏省
<Option>浙江省
<Option>安徽省
<Option>河南省
</select><BR>
<HR>
<Input Type="Submit" name="" value="提交">
<Input Type="reset" name="" value="复位">
</Form>
<pre>
<script language="JavaScript">
document.test9_1.elements[0].options[0].text="昆明市";
document.test9_1.elements[0].options[1].text="成都市";
</script>
</pre>
</Body>
</HTML>
```

第三个框架

主要作用是实现交互。

Test9_3.htm

```
<HTML>
<HEAD>
</HEAD>
<Body>
<Form name="test9_2">
请输入用户名：
<Input Type="text" name="text1" Value="" Size=20><BR>
<HR>
请选择:
<Input Type="Checkbox" name="checkbox1" Value="qb">全部信息<BR>
```

```
<Input Type="Checkbox" name="checkbox2" Value="bf">部分信息<BR>
<Input Type="Checkbox" name="checkbox3" Value="sy">所有城市<br>
<HR>
<Input Type="Submit" name="" value="提交">
<Input Type="reset" name="" value="复位">
<BR>
</Form>
<script language="JavaScript">
document.test9_2.elements[0].value="劳动和社会保障";
document.test9_2.elements[1].checked=true;
document.test9_2.elements[2].checked=true;
document.test9_2.elements[3].checked=false;
</script>
</Body>
</HTML>
```

在浏览器中的结果见图 9-2 所示。

图 9-2 在浏览器中结果

本讲介绍框架中的基本元素的主要功能和使用，利用 JavaScript 脚本可以非常方便、灵活地实现 Web 页面更为复杂的信息交互，这是 HTML 标识语言所不能具备的。从中可以看出 JavaScript 是多么的吸引众多的 Web 设计人员。