

# **JavaScript 简介**

**JavaScript** 是网景 ( Netscape ) 公司开发的一种基于客户端浏览器、面向 ( 基于 ) 对象、事件驱动式的网页脚本语言。 JavaScript 语言的前身叫作 Livescript。

- **JavaScript的特点：**
  1. 简单、易学、易用；
  2. 跨平台；IE、Navigator
  3. 符合ECMA ( 欧洲计算机制造协会 ) 标准，可移植；
  4. 事件驱动式的脚本程序设计思想；
  5. 动态、交互式操作方式。
- **JavaScript的作用：**
  1. 交互式操作；
  2. 表单验证；
  3. 网页特效；
  4. Web游戏
  5. 服务器脚本开发等。
- **JavaScript的编写环境：文本编辑器**
- **JavaScript的执行平台：Web浏览器**
- **JavaScript的执行方式：解释执行（由上而下）**
- **JavaScript的版本：JavaScript1.0——JavaScript1.4**
- **浏览器对JavaScript的支持：**
  1. JavaScript/IE3.0、JavaScript1.2/IE4.0；
  2. 微软允许用户自行设置对JavaScript处理模式。

- **JavaScript与Java、VBScript、JScript的关系：**

JavaScript与Java的区别体现在：

首先，它们是两个公司开发的不同的两个产品，Java是SUN公司推出的新一代面向对象的程序设计语言，特别适合于Internet应用程序开发；而JavaScript是Netscape公司的产品，其目的是为了扩展Netscape Navigator功能而开发的一种可以嵌入Web页面中的基于对象和事件驱动的解释性语言。

其次，JavaScript是基于对象的，而Java是面向对象的，即Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象。JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象和事件驱动的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用。

第三，两种语言在其浏览器中所执行的方式不一样。Java的源代码在传递到客户端执行之前，必须经过编译，因而客户端上必须具有相应平台上的仿真器或解释器，它可以通过编译器或解释器实现独立于某个特定的平台编译代码的束缚。JavaScript是一种解释性编程语言，其源代码在发往客户端执行之前不需经过编译，而是将文本格式的字符代码发送给客户，由浏览器解释执行。

第四，两种语言所采取的变量是不一样的。Java采用强类型变量检查，即所有变量在编译之前必须作声明。JavaScript中变量声明，采用其弱类型。即变量在使用前不需作声明，而是解释器在运行时检查其数据类型。

第五，代码格式不一样。Java是一种与HTML无关的格式，必须通过像HTML中引用外媒体那么进行装载，其代码以字节代码的形式保存在独立的文档中。JavaScript的代码是一种文本字符格式，可以直接嵌入HTML文档中，并且可动态装载。编写HTML文档就像编辑文本文件一样方便。

第六，嵌入方式不一样。在HTML文档中，两种编程

语言的标识不同，JavaScript使用<script>...</script>来标识，而Java使用<applet> ... </applet> 来标识。

第七，静态绑定和动态绑定。Java采用静态联编，即Java的对象引用必须在编译时的进行，以使编译器能够实现强类型检查，如不经编译则就无法实现对象引用的检查。JavaScript采用动态联编，即JavaScript的对象引用在运行时进行检查。

- **JavaScript的格式：**
  1. JavaScript区分大小写；
  2. JavaScript脚本程序须嵌入在HTML文件中；
  3. JavaScript脚本程序中不能包含HTML标记代码；（双引号）
  4. 每行写一条脚本语句；
  5. 语句末尾可以加分号；
  6. JavaScript脚本程序可以独立保存为一个外部文件，但其中不能包含<script></script>标签。

- **JavaScript脚本程序的几种基本格式：**

1. 

```
<script>
document.write("Hello World!!!");
</script>
```
2. 

```
<script language="JavaScript">
document.write("Hello World!!!");
</script>
```
3. 

```
<script language="JavaScript"
type="text/JavaScript">
document.write("Hello World!!!");
</script>
```
4. 

```
<script language="JavaScript1.2">
document.write("Hello World!!!");
</script>
```
5. 

```
<script src="hello.js"></script>
```

**注意：document.write("Hello World!!!"); 必须保存为一个外部文件：hello.js**

6. 

```
<script language="JavaScript">
```

```
<!--  
document.write("Hello World!!!");  
-->  
</script>
```

## 7. 使用JavaScript协议：

```
<a href="JavaScript:alert('Hello World!!!')">请  
单击</a>
```

```
<a href="#" onclick="alert('Hello World!!!')">  
请单击</a>
```

```
<a href="JavaScript://" onclick="alert('Hello  
World!!!')">请单击</a>
```

- **JavaScript的数据类型：**

1. 数值：整数、浮点数；
2. 逻辑值：布尔值；
3. 字符串值；
4. 空值；
5. 未定义值；

- **JavaScript数据的表示：**

1. 整数：由正负号、数字构成，八进制、十进制、十六进制；
2. 浮点数：由正负号、数字和小数点构成，常规记数法、科学记数法；
3. 逻辑值：true、false
4. 字符串值：单引号、双引号
5. 空值：null
6. 未定义值：根本不存在的对象、已定义但没有赋值的量；

- **JavaScript常量：**

1. 布尔常量：true      false
2. 整数常量：3721      0007      0xaff ( 0Xaff )
3. 浮点数常量：3.14      .001  
                  3.721e+3      -3.721E-3
4. 字符串常量："你的E-mail地址有误！"
5. 含转义字符的字符串常量：

转义字符	意义
\b	退格 ( Backspace )
\f	换页 ( Form feed )
\n	换行 ( New line )
\r	返回 ( Carriage return )
\t	制表符 ( Tab )
\'	单引号 ( ' )

\"	双引号 ("")
\\"\\	反斜线 (\ )

document.write("我爱\JavaScript\\")

document.write("文件在c:\\windows\\下")

document.write("<pre>未满十八岁\n不得进入！</pre>")

document.write("未满十八岁<br>不得进入！")

document.alert("密码不对\n请重新输入！")

## 6. 数组常量：

hobby=["听音乐","看电影"]  
hobby[0]、hobby[1]

hobby=["听音乐","","看电影",]  
hobby[0]、hobby[1]、hobby[2]、hobby[3]

## • JavaScript变量：

### 1. 变量命名规则：

- 首字符必须是大写或小写的字母或下划线（\_）或美元符（\$）；
- 后续的字符可以是字母、数字、下划线或美元符；  
变量名称不能是保留字；  
长度是任意；  
区分大小写；  
约定：集中置顶；  
使用局部变量；  
易于理解； stdId  
避免混乱。 username usrName

### 2. 声明变量：

- var stdId;
- var name,sex;
- var total=3721;
- var notNull=true;
- var name="李小龙" , sex="先生";
- var i=j=0;

### 3. 变量赋值 :

stdId = 2004007;

### 4. 变量作用域 : ( 方式、位置 )

- 全局变量 : 省略var , 或在函数外声明
- 局部变量 : 在函数内声明
- 全局变量可在整个脚本中被使用 , 可在不同的窗口中相互引用 ( 指定窗口名 )
- 例 :

```
<Script>
var langJS = "JavaScript"; //langJS是全局变量
    test();
```

```
function test() {
var langVBS = "VBScript"; //langVBS是局部
    变量
    document.write("<LI>" + langJS);
    document.write("<LI>" + langVBS);
    }
```

```
document.write("<LI>" + langJS);
document.write("<LI>" + langVBS);
</Script>
```



- **JavaScript表达式：**
  1. 算术表达式；
  2. 字符串表达式；
  3. 关系（比较）表达式；
  4. 逻辑表达式。

- **JavaScript运算符：**
  1. 根据处理对象的数目：
    - 单元运算符；
    - 二元运算符；
    - 三元运算符。

- 2. 根据功能：
  - 赋值运算符；

=      +=      -=      \*=      /=      %= ( 取余 )

- 算术运算符；

+      -      \*      /      % ( 取余 )      ++ ( 递增 )      -- ( 递减 ) -

例 1：

<Script>

var x = 11;

var y = 5;

```
with (document) {  
    write("x = 11, y = 5");  
    write("<LI>x + y 是 ", x + y);  
    write("<LI>x - y 是 ", x - y);
```

```

        write("<LI>x * y 是 ", x * y);
        write("<LI>x / y 是 ", x / y);
        write("<LI>x % y 是 ", x % y);
        write("<LI>++ x 是 ", ++ x);
        write("<LI>-- y 是 ", -- y);
    }
</Script>

```

例 2 :

<Script>

```
var x = y = 3;
```

```

with (document) {
    write("x = 3, y = 3 <br>");
    write("若 x = y++ 运算之后 : ");
    x = y++; //y → x , y+1 → y
    write("x 是 ", x, "; y 是 ", y, "<br>");
    write("再作 x = ++y 运算 : ");
    x = ++y; //y+1 → x , y+1 → y
    write("x 是 ", x, "; y 是 ", y);
}
</Script>

```

■ 字符串运算符 ;

+        +=

■ 比较运算符 ;

|    |    |               |                  |
|----|----|---------------|------------------|
| == | != | ==== ( 值及类型 ) | !== ( 值及<br>类型 ) |
| <  | <= | >             | >=               |

例 :

<Script>

```

var x = 5; //x 是数值5
var y = '5'; //y 是字符串5
var z = 6; //x 是数值6

```

```

        with (document) {
            write("x = 5, y = '5', z = 6");
            write("<LI>x == y 吗 ?", x == y);
            write("<LI>x === y 吗 ?", x === y);
            write("<LI>x != y 吗 ?", x != y);
            write("<LI>x == y 吗 ?", x == y);
            write("<LI>x <= z 吗 ?", x <= z);
            write("<LI>y <= z 吗 ?", y <= z);
            //类型自动转换
        }
    </Script>

```

■ 逻辑运算符；

&&      ||      !

例 1：

```

<Script>
var t = true;
var f = false;

```

```

with(document) {
    write("<OL><LI>true && true 的结果是 ", t &&
        t);
    write("<LI>true && false 的结果是 ", t && f);
    write("<LI>false && true 的结果是 ", f && t);
    write("<LI>false && false 的结果是 ", f && f);
    write("<LI>true && (1==1) 的结果是 ", t &&
        (1==1));
    write("<LI>false && 'A' 的结果是 ", f && 'A');
    write("<LI>'A' && false 的结果是 ", 'A' && f);
    write("<LI>true && 'A' 的结果是 ", t && 'A');
    write("<LI>'A' && true 的结果是 ", 'A' && t);
    write("<LI>'A' && 'B' 的结果是 ", 'A' && 'B');
    //&&：有一个不是逻辑值，只要第一个操作数的值为flase，则返回第一个操作数的值
    //否则，返回第二个操作数的值
}
```

```
    }  
</Script>
```

例 2 :

```
<Script>  
var t = true;  
var f = false;
```

```
with(document) {  
write("<OL><LI>true || true 的结果是 ", t || t);  
    write("<LI>true || false 的结果是 ", t || f);  
    write("<LI>false || true 的结果是 ", f || t);  
    write("<LI>false || false 的结果是 ", f || f);  
write("<LI>true || (1==1) 的结果是 ", t || (1==1));  
    write("<LI>false || 'A' 的结果是 ", f || 'A');  
    write("<LI>'A' || false 的结果是 ", 'A' || f);  
    write("<LI>true || 'A' 的结果是 ", t || 'A');  
    write("<LI>'A' || true 的结果是 ", 'A' || t);  
    write("<LI>'A' || 'B' 的结果是 ", 'A' || 'B');  
//|| : 有一个不是逻辑值，只要第一个操作数  
为的值true、字符或非零的数字，则返回第一  
个操作数的值，否则，返回第二个操作数的值  
}  
</Script>
```

例 3 :

```
<Script>  
with(document) {  
write("<LI>!true 的结果是 ", !true);  
write("<LI>!false 的结果是 ", !false);  
    write("<LI>!'A' 的结果是 ", !'A');  
    write("<LI>!0 的结果是 ", !0);}  
</Script>
```

- 逐位运算符；
- 特殊运算符。

1、new运算符：创建对象（实例）

格式：对象名称=new 对象类型（参数）

2、this运算符：表示当前对象

格式：this[.属性]

例：

```
<Script>
function validate(obj) {
    alert("你输入的值是：" + obj.value);
} </Script>
请输入任意字符：  

<INPUT TYPE="text" onKeyup="validate(this)">
```

3、条件运算符：三元运算符

格式：<条件表达式> ? 第一个值 : 第二个值

例：

```
NS = (document.layers) ? 1 : 0;
IE = (document.all) ? 1 : 0;
```

```
window.screen.width > 800 ?
imgheight=100:imgheight=100
window.screen.width > 800 ?
imgleft=15:imgleft=122
```

```
<Script>
function showSex() {
onOroff = document.forms[0].sex[0].checked
status = (onOroff)? "帅哥" : "美女"
alert("Hello! " + status)
}
</Script>
```

请输入你的性别：  
<FORM onClick=showSex()>

```
<INPUT TYPE=radio NAME=sex>男生  
<INPUT TYPE=radio NAME=sex>女生  
</FORM>
```

### 3. 运算符执行的优先顺序：

类型	运算符
括号	()
一元	! ~ - ++ -- typeof void delete
算术	* / + -
位位移	<< >> >>>
比较	< <= > >= == !=
位逻辑	& ^ ( xor )
逻辑	&&
三元条件	?
赋值	= += -= *= /= %= <<= >>= >>>= &= ^=  =

- **JavaScript语句：**

1. **注释语句：**

- **单行注释：//注释文字**
- **多行注释：/\*  
注释文字  
\*/**

- **例：**

```
*****  
*      源码之家      *  
*      http://www.mycodes.net      *  
*      下载:http://www.mycodes.net      *  
*      论坛http://www.mycodes.net/bbs      *  
*****
```

2. **with语句：(对象操作语句)**

- **功能：为一段程序建立默认对象。**

- **格式：**

```
with (<对象>){  
    <语句组>  
}
```

- **例 1：**

```
with (document) {  
    write ("限时抢购物品 : ");  
    write ("<Li>ViewSonic 17\" 显示器。 ");  
    write ("<Li>EPSON 打印机。 ");  
}
```

- **例 2：**

```
document.write ("限时抢购物品 : ");  
document.write ("<Li>ViewSonic 17\" 显示  
器。 ");  
document.write ("<Li>EPSON 打印机。 ");
```

3. if...else语句：

格式1：

```
if (<表达式>)
    <语句1>;
    else
        <语句2>;
```

```
if (<表达式>) <语句1>;
    else <语句2>;
```

例：

```
<Script>
    var now = new Date();
    var hour = now.getHours();

    if (6 < hour && hour < 18)
        document.write ("辛勤工作才能快乐收割！");
        else
            document.write ("休息一下，充电后再出发。");
    </Script>
```

格式2：

```
if (<表达式>){
    <语句组1>
    }else{
        <语句组2>
    }
```

格式3：

```
if (<表达式1>){  
    <语句组1>  
}else if (<表达式2>){  
    <语句组2>  
    }else{  
        <语句组3>  
    }
```

例1：

```
<Script>
var now = new Date();
var day = now.getDay();
var dayName;

if (day == 0) dayName = "星期日";
else if (day == 1) dayName = "星期一";
else if (day == 2) dayName = "星期二";
else if (day == 3) dayName = "星期三";
else if (day == 4) dayName = "星期四";
else if (day == 5) dayName = "星期五";
else dayName = "星期六";

document.write ("今天是快乐的", dayName);
</Script>
```

例2：

```
<Script>
var now = new Date();
var day = now.getDay();
var dayName;
```

```
if (day = 0) dayName = "星期日";
else if (day = 1) dayName = "星期一";
else if (day = 2) dayName = "星期二";
else if (day = 3) dayName = "星期三";
else if (day = 4) dayName = "星期四";
else if (day = 5) dayName = "星期五";
else dayName = "星期六";
```

```
document.write ("今天是快乐的", dayName);
</Script>
```

4.                  while语句：

格式1：

```
while (<表达式>
     语句；
```

格式2：

```
while (<表达式>){
     <语句组>
     }
```

例：
<Script>

```
var i = 5;  
  
while ( i > 0 ) {  
    document.write("i = " ,i ,"<BR>");  
    i--;  
}  
  
</Script>
```

5. do...while语句：

格式：

```
do {  
    <语句组>  
} while (<表达式>)
```

例：  
<Script>  
var i = 5;

```
do {  
    document.write("i = " ,i ,"<BR>");  
    i--;
```

```
 } while ( i > 0 )
```

```
</Script>
```

6. for语句：

格式：

```
for (<初始表达式> ; <条件表达式> ; <变动量表达式>){  
    <语句组>  
}
```

例：

```
<Script>  
for ( var i = 5; i > 0; i-- ) {  
    document.write("i = " ,i , "<BR>");  
}
```

```
</Script>
```

7. for...in语句：重复执行指定对象的所有属性

格式：

```
for ( 变量 in 对象 ){
    <语句组>
}
```

例：

```
<Script>
function member(name, sex) { //构造函数member
    this.name = name;
    this.sex = sex;
}
```

```
function showProperty(obj, objString) {
    var str = "";
    for (var i in obj)
        str += objString + "." + i + " = " + obj[i] + "<BR>";
    return str;
}
```

```
papa = new member("杨宏文", "男生");//建立对象实例
papa
document.write(showProperty(papa, "papa"))
```

```
</Script>
```

8. break语句：

格式：break

例：

```
<Script>
var i = 5;
while ( i > 0 ) {
    if ( i == 3 ) break;
    document.write("i = " ,i , "<BR>");
    i--;
}
```

```
</Script>
```

9. continue语句：

格式：continue

例：

```
<Script>
  var i = 5;
  while ( i > 0 ) {
    i--;
    if ( i == 3 ) continue;
    document.write("i = " ,i ,"  
");
  }
```

```
</Script>
```

10. switch语句：

格式：

```
switch (<表达式>) {
  case <数值1>:<语句组1>
    break;
  case <数值2>:<语句组2>
    break;
  ...
  default <语句组>
}
```

例：

```
<Script>
function greet(givenSex) {
    switch(givenSex) {
        case "帅哥" : alert("你好啊！");
        break;
        case "美女" : alert("你好啊！");
        break;
    }
}
</Script>
```

请输入性别：

```
<FORM>
<INPUT TYPE="radio" NAME="sex"
```

```
onClick="greet('帅哥')"
帅哥
<INPUT TYPE="radio" NAME="sex"
onClick="greet('美女')"
美女
</FORM>
```

- 对象：  
对象是一组具有属性和方法的经过组织的数据。
- 默认对象：  
1. 日期对象；（日期基线：1970年1月1日00：00：00

建立日期对象(实例)：

格式：日期对象名称=new Date([日期参数])

- 日期参数：1.省略；  
 2.英文-数值格式：月 日，公元年 [时 : 分]  
     如：today=new Date("October 1,2008 1  
 3.数值格式：公元年，月，日，[时，分]  
     如：today=new Date(2008,10,1

日期对象的方法：

格式：日期对象名称.方法 ([参数])

获取当地时间：

getYear()	返回年份数
getFullYear()	返回年份数
getMonth()	返回月份数 (0--11)
getDate()	返回日期数 (1--31)
getDay()	返回星期数 (0--6)
getHours()	返回时数 (0--23)
getMinutes()	返回分数 (0--59)
getSeconds()	返回秒数 (0--59)
getMilliseconds()	返回毫秒数 (0--999)
getTime()	返回对应日期基线的毫秒
Date.parse(日期字串)	返回对应日期基线的毫秒
setTime(时间值)	指定一日期对象的值
toGMTString()	以GMT格式表示日期对象

toUTCString()

以GMT格式表示日期对象

2.

数组对象；

建立数组对象：

格式 1：数组对象名称=new Array([元素个数])

格式 2：数组对象名称=new Array([[元素1][，元素2]...])

格式 3：数组对象名称=[元素1[，元素2，...]]

例：

fruit=new Array(3); // fruit=new Array();

fruit[0]="苹果";

fruit[1]="梨子";

fruit[2]="橘子";

fruit=new Array("苹果"， "梨子"， "橘子")

fruit=["苹果"， "梨子"， "橘子"];

数组对象的属性：

格式：数组对象名称.属性

属性：length      数组长度

例：

<Script>

```
var fruit = new Array("苹果", "梨子", "橘子");
var i;
```

```
for (i=0; i < fruit.length; i++) {
```

```
document.write("fruit [", i, "] = ", fruit[i], "<br>")  
}
```

</Script>

## 数组对象的方法：

格式：数组对象名称.方法([参数])

### 方法：

join([分隔符])	数组元素组合为字符串
toString()	以字符串表示数组
reverse()	数组反转
valueOf()	返回数组值

### 例：

<Script>

```
var fruit = new Array("苹果", "梨子", "橘子");
```

```
document.write("<LI>", fruit.join());  
document.write("<LI>", fruit.join("、 "));  
document.write("<LI>", fruit.toString());  
document.write("<LI>", fruit.reverse().join());  
document.write("<LI>", fruit.valueOf());
```

</Script>

## 二维数组：

### 例：

<Script>

```
var fruit = new Array(3);  
fruit[0] = new Array("苹果", 2);  
fruit[1] = new Array("梨子", 4);  
fruit[2] = new Array("橘子", 2);  
  
for (i=0; i < fruit.length; i++) {  
    for (j=0; j < fruit[0].length; j++) {  
        document.write("fruit [", i, "][", j, "] = ", fruit[i][j], "<br  
    }  
}
```

```
document.write("<br>");  
}
```

```
</Script>
```

3.      **字符串对象**；

建立字符串对象：

格式：字符串对象名称=new String(字符串常量)  
      格式：字符串变量名称="字符串常量"

字符串对象的属性：

格式：字符串对象名称.属性

属性：length      字符串长度

字符串对象的方法：

格式：字符串对象名称.方法

方法：

|                     |                   |
|---------------------|-------------------|
| bold()              | 粗体                |
| italics()           | 斜体                |
| strike()            | 删除线               |
| fontsize(字级大小)      | 文字大小              |
| fontcolor(#rrggbb)  | 文字颜色              |
| sup()               | 上标                |
| sub()               | 下标                |
| toUpperCase()       | 大写                |
| toLowerCase()       | 小写                |
| charAt(索引)          | 返回索引位置的字符         |
| charCodeAt(索引)      | 返回索引位置的ASCII字符码，示 |
| indexOf("字符串"[,索引]) | 返回字符串在对象中的索引位置    |
|                     | 返回字符串在对象中的索引位置    |

|               |                         |                    |
|---------------|-------------------------|--------------------|
|               | lastIndexOf("字符串"[,索引]) | 索 )                |
|               | search("字符串")           | 返回字符串在对象中的索引位置     |
|               | replace("字符串1","字符串2")  | 字符串2替换字符串1         |
|               | slice(索引i[,索引j])        | 返回索引i到索引j-1的子串     |
|               | split(["字符串"][,限制])     | 将字符串从对象中删除         |
|               | substr(start[,length])  | 返回特定长度的字符串         |
|               | substring(索引i[,索引j])    | 返回索引i到索引j-1的子串     |
|               | link("url")             | 设置链接               |
| match(/匹配字符/) | \d                      | 匹配一个数字字符。          |
|               | \D                      | 匹配一个非数字字符。         |
|               | \n                      | 匹配一个换行符。           |
|               | \r                      | 匹配一个回车符。           |
|               | \s                      | 匹配一个空格符。           |
|               | \S                      | 匹配任意非空格字符。         |
|               | \t                      | 匹配一个tab符。          |
|               | \W                      | 匹配任何非单词字符。         |
|               | \num                    | 匹配正整数num。          |
|               | /n/                     | 匹配八进制，十六进制的escape值 |

|            |        |
|------------|--------|
| toString() | 返回字符串  |
| valueOf()  | 返回字符串值 |

例1：

```
<Script>
var str = "JavaScript";

document.write(str.bold(),"<BR>");
document.write(str.fixed(),"<BR>");
document.write(str.fontcolor("red"),"<BR>");
document.write(str.fontsize(5),"<BR>");

</Script>
```

例2：

```
<Script>

var str = "JavaScript";
var num = 1234;

document.write(str.toUpperCase(), "<BR>");
document.write(num.toString().charAt(2),"<BR>");
document.write(str.substring(0,4), "<BR>");

</Script>
```

例3：

```
<Script>

var str = "JavaScript";

document.write(str, " 有 ", str.length, " 个字<BR>");
document.write(str.fontcolor("green"), " 不是 ");
document.write(str.substr(0,4).fontcolor("red"));
document.write(" 也不是 ", str.replace("Java", "VB"))

</Script>
```

例4：  
<Script>

```
var str = "hubert@abc.com.cn";
var idx = str.search("@");
var usr = str.substr(0,idx);

document.write(usr.toUpperCase().fontsize(7), "<BR>";
document.write("你的电子信箱是", str);

</Script>
```

例5：  
<Script>

```
function isEmail(){
var str = document.form1.add.value;
if (str.indexOf "@" == -1)
alert("请填写正确的EMail地址");
else
alert("OK!");
}
```

</Script>

```
<FORM name="form1">
请输入你的 EMail 地址：
<INPUT TYPE="text" name="add">
<INPUT TYPE="button" value="开始检查" onClick="isEm
</FORM>
```

4. 布尔对象；

建立布尔对象：

格式：布尔对象名称=new Boolean(转换值)  
转换值：null、未定义、0、或false均转换成false

布尔对象的方法：

格式：布尔对象名称.方法

方法：toString()

valueOf()

例：

<Script>

```
x = new Boolean();
y = new Boolean(true);
z = new Boolean(0);
```

```
document.write(x, "<br>");
document.write(y, "<br>");
document.write(z, "<br>");
```

</Script>

## 5. 数学对象；（静态对象）

数学对象的属性：

格式：Math.属性

属性：

E	自然对数的底数
LN2	2的自然对数
LN10	10的自然对数
LOG2E	以2为底e的对数
LOG10E	以10为底e的对数
PI	圆周率
SQRT1_2	1/2的平方根
SQRT2	2的平方根

数学对象的方法：

格式：Math.方法（参数）

### 方法：

ceil(数值)	大于等于该数值的最小整数
floor(数值)	小于等于该数值的最大整数
min(数值1,数值2)	最小值
max(数值1,数值2)	最大值
pow(数值1,数值2)	数值1的数值2次方
random()	0到1的随机数
round(数值)	最接近该数值的整数
sqrt(数值)	开平方根
abs、sin(弧度)、cos、tan、asin、acos、atan、exp、log	

### 例1：

```
<Script>
```

```
    with (document) {
        write("<LI>2 的平方根值是 ", Math.sqrt(2));
        write("<LI>2 <sup>3</sup> = ", Math.pow(2,3));
        write("<LI>最接近 3.14 的整数是 ", Math.round(3.14));
    }
```

```
</Script>
```

### 例2：

```
<Script>
```

```
    var now = new Date();
    var firstDay = new Date("Nov 10, 1999");
    var duration = now - firstDay;
    var msPerDay = 24 * 60 * 60 * 1000; //换算成毫秒
    days = Math.round(duration/msPerDay);

    document.write("本网站已经开幕" + days + "天了")
```

```
</Script>
```

### 例3：

```
<Script>
```

```
    var promote = new Array(3);
    promote[0] = "拍卖区又有新货到了，赶快来捡便宜啊。
    promote[1] = "成为会员，马上享受八折优惠，还可以参加
    promote[2] = "庆祝访问人数突破十万人次，填问卷就送大奖
                啊！";
```

```
    index = Math.floor(Math.random() * promote.length);
    document.write(promote[index]);
```

```
</Script>
```

- 6. 数值对象；
- 7. 函数对象；
- 8. 自定义对象。

- 自定义对象；
- 构造函数定义对象类型；
- 建立对象实例。
- 例：

```
<Script>
```

```
function member(name, sex) {
    this.name = name;
    this.sex = sex;
}
```

```
var papa = new member("杨宏文", "男生");
var mama = new member("黄雅玲", "女生");
var doggy = new member("奇 奇", "宠物狗");
```

```
    document.write(papa.name);
    document.write("是", papa.sex);
```

```
</Script>
```

- 动态的定义对象属性；

例：

为指定的对象实例定义属性：papa.hobby="看电视";

为对象定义属性：member.prototype.hobby=null;

papa.hobby="上网";

mama.hobby="逛街";

doggy.hobby="啃骨头";

■ 定义对象的方法；

■ 构造函数定义对象方法名；

■ 建立方法的描述函数。

例：

<Script>

```
function member(name, sex) {  
    this.name = name;  
    this.sex = sex;  
    this.display = display;  
}
```

```
function display() {  
var str = this.name + "是" + this.sex;  
document.write("<LI>" + str);  
}
```

```
var papa = new member("杨宏文", "男生");  
var mama = new member("黄雅玲", "女生");  
var doggy = new member("奇 奇", "宠物狗");
```

```
papa.display();  
mama.display();  
doggy.display();
```

</Script>

## 9. 利用对象原型（prototype）为默认对象定义属性：

<Script>

```
String.prototype.replaceAll = strReplace;
```

```
function strReplace(findText, replaceText) {  
    var str = new String(this);  
    while (str.indexOf(findText)!=-1) {  
        str = str.replace(findText, replaceText);  
    }  
    return str;  
}  
  
myStr = "告诉你 - 如果你正在寻找一本能帮助你彻底研  
    "JavaScript的书籍，请你一定要认明暮峰的" +  
    "JavaScript教学范本，让你事半功倍，功力大增。'  
document.write("<LI>原稿是 : <BLOCKQUOTE>" + my  
                "</BLOCKQUOTE>");  
document.write("<LI>利用Replace()将「你」改成「您」  
+"<BLOCKQUOTE>" + myStr.replace('你','您') + "</BLOCKQ  
document.write("<LI>利用自定义的字符串方法 - 全部  
+"<BLOCKQUOTE>" + myStr.replaceAll('你','您') +  
                "</BLOCKQUOTE>");  
  
</Script>
```

- 

默认函数：

1. 编码函数 escape()：将非字母、数字字符转换成ASCII码

例：[sample/unescape](#)

2. 译码函数 unescape()：将ASCII码转换成字母、数字字符

例：[sample/unescape](#)

3. 求值函数 eval()：

格式：eval(<表达式>)

例1：字符串运算  
[`<Script>`](#)

```
x = 1 + 2;  
y = "1 + 2";  
z = eval("1 + 2");
```

```
document.write("<LI>1 + 2 = ", x);  
document.write("<LI>\\"1 + 2\\" = ", y);  
document.write("<LI>eval(\\"1 + 2\\") = ", z);
```

[`</Script>`](#)

例2：对象操作  
[`<Script>`](#)

```
function show(obj){  
    var  
    str=eval("document.form."+obj+".value");  
    alert(str);
```

```
        }
    </Script>

<form name="form" id="form">
    姓名：
    <input name="name" type="text" id="name">
    <input type="button" name="Button"
    value="Button" onclick=show("name")>
</form>
```

#### 4. 数值判断函数 isNaN() : 是否为数值

格式 : isNaN(<量>)

例 :

```
<Script>
```

```
var x = 15;
var y = "黄雅玲";
```

```
document.write("<LI>x 不是数值
    吗 ? ", isNaN(x));
document.write("<LI>y 不是数值
    吗 ? ", isNaN(y));
```

```
</Script>
```

#### 5. 整数转换函数 parseInt() : 将不同进制 (二、八、十六) 的数值转换成十进制整数

格式 : parseInt(数值字符串[, 底数])

底数省略，则按内容转换：

0x	0X	十六进制
0		八进制
其它		十进制

例 :

```
<Script>
```

```
// 二进位转成十进位  
  
document.write("1101<sub>2</sub> = "  
,parseInt("1101", 2),"<sub>10</sub><br>");  
  
// 十六进位转成十进位  
  
document.write("BFFF<sub>16</sub> = "  
,parseInt("BFFF", 16),"<sub>10</sub><br>");  
  
</Script>
```

## 6. 浮点数转换函数 `parseFloat()`：将数值字符串转换成浮点数

格式：`parseFloat(数值字符串)`

例：  
<Script>

```
document.write(parseInt("3.1234A56"), "  
    <br>");  
document.write(parseFloat("3.1234A56"), "  
    <br>");  
  
</Script>
```

- **函数：**  
独立于主程序的、具有特定功能的一段程序代码块。
- **函数的定义：**  
**格式：**  
`function 函数名([参数[,参数...]]){  
 <语句组>  
 [return <表达式> ;]  
}`
- **约定：**  
1、**函数名**：易于识别；（同变量命名规则）  
2、**程序代码**：模块化设计；  
3、**函数位置**：按逻辑顺序，集中置顶。（<head>...</head>）
- **return语句：**  
**格式：**  
`return <表达式> ;`  
**功能**：返回表达式的值。
- **函数的调用：**  
**格式：**  
`函数名([参数[,参数...]])`  
**例1：**  
`<Script>`  
  
`function showName(name){  
document.write ( "我是" + name );  
}  
  
showName("黄雅玲");`

```
</Script>
```

例2：  
<Script>

```
function showName(name){  
    str="我是" + name;  
    return str;  
}
```

```
document.write(showName("黃雅玲"));
```

```
</Script>
```

- 事件：

用户对浏览器所做的特定的动作（操作），是实现交互操作的一种机制。

| 事件名称   | 适用对象           | 意义      | 说明             |
|--|----------------|---------|----------------|
| Abort  | image          | 终止      | 当图形尚未完全加载前     |
| Blur   |                | 失去焦点    |                |
| Change   | t/pw/ta/select | 改变      |                |
| DragDrop   | window         | 拖曳      |                |
| Error  |                | img/win | 错误加载文件或图形时发生错误 |
| Focus  |                | 取得焦点    |                |
| Move   | window         | 移动      |                |
| Reset  | form           | 重置      |                |
| Submit   | form           |         |                |
| Click/DblClick、KeyDown/KeyPress/KeyUp、<br>Load/Unload、<br>MouseDown/MouseUp/MouseOver/MouseOut/MouseMove |                |         |                |

- 事件处理程序：  
浏览器响应某个事件，实现用户的交互操作而进行的处理（过程）。
- 事件处理程序的调用：  
浏览器等待用户的交互操作，并在事件发生时，自动调用事件处理程序（函数），完成事件处理过程。

### HTML标签属性：

格式：

<tag on事件=<语句组>|<函数名>>

例1：

```
<body onload="alert('建议浏览器的分辨率：  
800x600');">  
<body onload="var str='建议浏览器的分辨率：  
800x600';alert(str);">
```

例2：

```
<script>  
function show(){  
var str="建议浏览器的分辨率：800x600";  
alert(str);  
}  
</script>  
<body onload="show();">
```

### 对象属性：

格式：

对象名.on事件=<语句>|<函数名>

例1：

```
document.onload=alert("建议浏览器的分辨率：
```

```
800x600");
```

```
var str="建议浏览器的分辨率 : 800x600";  
document.onload=alert(str);
```

例2 :

```
<script>  
function show(){  
var str="建议浏览器的分辨率 : 800x600";  
alert(str);  
}  
document.onload=show();  
</script>
```

例1 :

```
<Body>  
<FORM>  
请输入基本资料 : <BR>  
姓名 :  
<INPUT TYPE="text" NAME="usr" SIZE="8">  
<INPUT TYPE="button" VALUE=" 请单击"  
onClick="alert('谢谢你的填写...')">  
</Body>
```

例2 :

```
<Script>  
function handelError(img){  
msg = "有一图文件 , 名为 : \\" + img.name + "\\n无法顺  
利显示 , 请通知系统管理人员" + " , 敬备薄礼相送。";  
alert(msg);  
}  
</Script>
```

```
<IMG SRC="abc.gif" NAME="中国电信的广告"  
onError="handelError(this)">
```

例3 :

```
<Body>  
<A HREF="http://www.hubert.idv.tw/"
```

```
onMouseOver="status='最棒的学习网站';return true;"  
onMouseOut="status='完毕'">文哥网络技术学习网</A>  
</Body>
```

例4：

```
<Body>  
    <FONT STYLE="cursor:hand"  
        onClick="location='http://www.hubert.idv.tw/'"  
        onMouseOver="status='最棒的在线学习网站';  
        this.color='red';return true;" onMouseOut="status='完毕';  
        this.color='blue';">文哥网络技术学习网</FONT>  
</Body>
```

例5：

```
<Script>  
function mOver(object,msg){  
    status = msg;  
    object.color = "red";  
    object.face = "华文楷体";  
}  
  
function mOut(object){  
    status = '完毕';  
    object.color = "blue";  
    object.face = "幼圆";  
}  
</Script>  
  
<Body>  
    <FONT STYLE="cursor:hand"  
        onClick="location='http://www.hubert.idv.tw/'"  
        onMouseOver="mOver(this,'最棒的线上学习网站'); return  
        true;" onMouseOut="mOut(this)">文哥网络技术学习网  
    </FONT>  
</Body>
```

例6：

```
<STYLE> A {text-decoration:none} </STYLE>
```

```
<BODY>
搜尋引擎 : <BR>
```

```
<IMG SRC="images\snow1.gif" NAME=gif_1>
<A HREF="http://www.yam.com/"
onMouseOver="document.gif_1.src='images\\snow.gif'"
onMouseOut="document.gif_1.src='images\\snow1.gif'">蕃薯
藤</A><BR>
```

```
<IMG SRC="images\snow1.gif" NAME=gif_2>
<A HREF="http://www.kimo.com.tw/"
onMouseOver="document.gif_2.src='images\\snow.gif'"
onMouseOut="document.gif_2.src='images\\snow1.gif'">奇摩
站</A>
</BODY>
```

例7 :

```
<Script>
```

```
var url = new Array(3);
url[0] = "http://www.yam.org.tw/";
url[1] = "http://www.kimo.com/";
url[2] = "http://chinese.yahoo.com/";
```

```
function goto(i) {
    location = url[i];
}
```

```
</Script>
```

```
<table width=250><tr><td>
<form><fieldset>
<legend>搜尋引擎</legend>
<input name="go" type="radio" onClick="goto(0)">蕃薯藤
<input name="go" type="radio" onClick="goto(1)">奇摩
<input name="go" type="radio" onClick="goto(2)">中文雅虎
</fieldset></form>
</td>
</tr>
</table>
```



- 定时器：（延迟器）

用以指定在一段特定的时间后执行某段程序。

- **setTimeout()**：（1.0版）

格式：

[定时器对象名=] setTimeout(“<表达式>” , 毫秒)

功能：执行<表达式>一次。

例1：

<Script>

```
function count() {  
    setTimeout("alert('三秒到了')",3000)  
}
```

</Script>

```
<INPUT TYPE="button" VALUE=" 计时开始"  
       onClick="count()">
```

例2：

<Script>

```
function show() {  
    document.all['news'].style.display = "";  
    setTimeout("hide()",500);  
}
```

```
function hide() {  
    document.all['news'].style.display = "none";  
    setTimeout("show()",500);  
}
```

```
</Script>

<Body onload="show()">
最新消息 : <FONT ID="news" STYLE="display:none">
    十面埋伏...</FONT>
</Body>
```

- **clearTimeout()** : 终止定时器

格式 :

clearTimeout(定时器对象名)

- **setInterval()** : ( 1.2 版 )

格式 :

[定时器对象名 = ] setInterval(" <表达式> " , 毫秒)

功能 : 重复执行<表达式> , 直至窗口、框架被关闭或  
执行 clearInterval。

- **clearInterval()** : 终止定时器

格式 :

clearInterval(定时器对象名)

例1 :

```
<Script>
```

```
var sec = 0;
timerID = setInterval("count()",1000);
```

```
function count() {
```

```
    num.innerHTML = sec++;
}
```

```
</Script>
```

停留时间：

```
<FONT ID="num" FACE="impact">0</FONT>秒钟
<INPUT TYPE="button" VALUE="停止"
onClick="clearInterval(timerID)">
```

例2：

```
<Script>
```

```
var str = "这是一个在线拍卖的网站，请尽情血拼
吧！";
var seq = 0;
```

```
function scroll() {
msg = str.substring(0, seq+1);
banner.innerHTML = msg;
seq++;
if (seq >= str.length) seq = 0;
}
```

```
</Script>
```

```
<Body onLoad="setInterval('scroll()',500)">
<FONT ID="banner"></FONT>
</Body>
```

- 图像对象：  
网页中的图像均会被自动看作图像对象，并依顺序，分别表示为document.images[0]，document.images[1]...
  - 建立图像对象：  
格式：  
图像对象名称=new Image([宽度],[高度]) //px
  - 图像对象的属性：  
border complete height hspace lowssrc name src vspace width
  - 图像对象的事件：  
onAbort onError onKeyDown onKeyPress onKeyUp  
onLoad
- 例1：（预载）  
<Script>
- ```
img0 = new Image();
img0.src = "images/snow0.gif";

img1 = new Image();
img1.src = "images/snow1.gif";

document.write ("已经读取两个图文件，但此时不显示。");

</Script>
```
- 例2：  
<Script>
- ```
function img-preload(idx){
```

```
eval("img"+idx+" = new Image()");
eval("img"+idx+".src = 'images/snow"+idx+".gif'");
}

img-preload(0);
img-preload(1);
document.write ("已经读取两个图文件，但此时不显示。");
```

</Script>

例3：

```
<Script>
function img-preload(imgname,idx){
    eval("img"+idx+" = new Image()");
    eval("img"+idx+".src = 'images/"+imgname+".gif'");
}

img-preload("snow0",0);
img-preload("snow1",1);
document.write ("已经读取两个图文件，但此时不显示。");
```

</Script>

- **Navigator对象：(领航员)**

检测浏览器的版本、所支持的MIME类型、已安装的外挂程序（plug-in）。该对象包含两个子对象：外挂对象、MIME类型对象。

- **Navigator对象的属性：**  
格式：navigator.属性

appCodeName	代码
appName	名称
appVersion	版本
language	语言
mimeType	以数组表示所支持的MIME类型
platform	编译浏览器的机器类型
plugins	以数组表示已安装的外挂程序
userAgent	用户代理程序的表头

例1：  
<Script>

```
with (document) {  
    write ("你的浏览器信息 : <OL>");  
    write ("<LI>代码 :" + navigator.appCodeName);  
    write ("<LI>名称 :" + navigator.appName);  
    write ("<LI>版本 :" + navigator.appVersion);  
    write ("<LI>语言 :" + navigator.language);  
    write ("<LI>编译平台 :" + navigator.platform);  
    write ("<LI>用户表头 :" + navigator.userAgent);  
}
```

</Script>

例2：

```

<Script>

    if (document.all) {
        document.write("你的浏览器是 : MSIE");
    } else {
        document.write("你的浏览器是 : Navigator");
    }

</Script>

```

- **plugin对象的属性 :**

格式 : navigator.plugins.属性

description	外挂程序模块的描述
filename	外挂程序模块的文件名
length	外挂程序模块的个数
name	外挂程序模块的名称

例 :

```

<Script>

var len = navigator.plugins.length;
with (document) {
    write ("你的浏览器共支持" + len + "种plug-in :
            <BR>");
    write ("<TABLE BORDER>")
    write ("<CAPTION>PLUG-IN 清单</CAPTION>")
    write ("<TR><TH> <TH>名称<TH>描述<TH>文件
          名")
    for (var i=0; i<len; i++)
        write("<TR><TD>" + i +
              "<TD>" + navigator.plugins[i].name +
              "<TD>" + navigator.plugins[i].description +
              "<TD>" + navigator.plugins[i].filename);
}

```

}

</Script>

- **mimeType**对象的属性：

格式：navigator.mimeTypes.属性

description	MIME类型的描述
enablePlugin	对应到哪个外挂模块
length	MIME类型的数目
suffixes	MIME类型的扩展名
type	MIME类型的名称

例：

<Script>

```
var len = navigator.mimeTypes.length;
with (document) {
    write ("你的浏览器共支持" + len + "种MIME类
型 : ");
    write ("<TABLE BORDER>")
    write ("<CAPTION>MIME type 清单</CAPTION>")
    write ("<TR><TH> <TH>名称<TH>描述<TH>扩展
名<TH>附注")
    for (var i=0; i<len; i++) {
        write("<TR><TD>" + i +
        "<TD>" + navigator.mimeTypes[i].type +
        "<TD>" + navigator.mimeTypes[i].description +
        "<TD>" + navigator.mimeTypes[i].suffixes +
        "<TD>" +
        navigator.mimeTypes[i].enabledPlugin.name);
    }
}
```

</Script>



- 窗口对象的属性和方法：

格式：

[window.]属性  
[window.]方法（参数）

opener.属性  
opener.方法（参数）  
self.属性  
self.方法（参数）  
parent.属性  
parent.方法（参数）  
top.属性  
top.方法（参数）

窗口名称.属性  
窗口名称.方法（参数）

- 窗口对象的属性：

document	当前文件的信息
location	当前URL的信息
name	窗口名称
status	状态栏的临时信息
defaultStatus	状态栏默认信息
history	该窗口最近查阅过的网页
closed	判断窗口是否关闭，返回布尔值
opener	open方法打开的窗口的源窗口
outerHeight	窗口边界的垂直尺寸，px
outerWidth	窗口边界的水平尺寸，px
pageXOffset	网页x-position的位置
pageYOffset	网页y-position的位置
innerHeight	窗口内容区的垂直尺寸，px
innerWidth	窗口内容区的水平尺寸，px

screenX	窗口左边缘的X坐标
screenY	窗口上边缘的Y坐标
self	当前窗口
top	最上方的窗口
parent	当前窗口或框架的框架组
frames	对应到窗口中的框架
length	框架的个数
locationbar	浏览器地址栏
menubar	浏览器菜单栏
scrollbars	浏览器滚动条
statusbar	浏览器状态栏
toolbar	浏览器工具栏
offscreenBuffering	是否更新窗口外的区域
personalbars	浏览器的个人工具栏，仅Navigator

- 窗口对象的方法：

alert(信息字符串)	弹出警告信息
confirm(信息字符串)	显示确认信息对话框
prompt(提示字符串[ , 默认值])	显示提示信息，并提供可输入的字段
atob(译码字符串)	对base-64编码字符串进行译码
btoa(字符串)	将进行base-64编码
back()	回到历史记录的上一网页
forward()	加载历史记录中的下一网页
open(URL , 窗口名称[ , 窗口规格])	
focus()	焦点移到该窗口
blur()	窗口转成背景
stop()	停止加载网页
close()	
enableExternalCapture()	允许有框架的窗口获取事件

disableExternalCapture()	关闭enableExternalCapture()
captureEvents(事件类型)	捕捉窗口的特定事件
routeEvent(事件)	传送已捕捉的事件
handleEvent(事件)	使特定事件的处理生效
releaseEvents(事件类型)	释放已获取的事件
moveBy(水平点数 , 垂直点数)	相对定位
moveTo(x坐标 , y坐标)	绝对定位
setResizable(true false)	是否允许调整窗口大小
resizeBy(水平点数 , 垂直点数)	相对调整窗口大小
resizeTo(宽度 , 高度)	绝对调整窗口大小
scroll(x坐标 , y坐标)	绝对滚动窗口
scrollBy(水平点数 , 垂直点数)	相对滚动窗口
scrollTo(x坐标 , y坐标)	绝对滚动窗口
setInterval(表达式 , 毫秒)	
setTimeout(表达式 , 毫秒)	
clearInterval(定时器对象)	
clearTimeout(定时器对象)	
home()	进入浏览器设置的主页
find([字符串 [,caseSensitivr,backward]]])	查找窗口中特定的字符串
print()	
setHotKeys(true false)	激活或关闭组合键
setZOptions()	设置窗口重叠时的堆栈顺序

- 窗口对象的事件处理程序：

onBlur onDragDrop onError onFocus onLoad onMove onResize  
onUnload

例1：  
<Script>

```
function checkPassword(testObject) {  
    if (testObject.value.length < 4) {  
        alert("密码长度不得小于四");  
        testObject.focus();  
        testObject.select();  
    }  
}
```

```
</Script>
```

请输入密码：

```
<INPUT TYPE="text" onBlur="checkPassword(this)">
```

例2：

```
<Script>
```

```
if (confirm("你满十八岁了吗?"))  
    location = "adult.htm";  
    else  
        alert("等你成年以後再来吧!");
```

```
</Script>
```

例3：

```
<Script>
```

```
var bgColor =  
prompt("你喜欢哪一种底色 : \n浅蓝色请按1 , 粉红色请按2",1)  
  
if (bgColor == 1) document.bgColor = "#CCFFFF";  
else if (bgColor == 2) document.bgColor = "#FFCCFF";  
else document.bgColor = "#FFFFFF";
```

```
</Script>
```

例4：

<Script>

```
function grow() {  
    resizeBy(0, 50);  
}
```

```
function shrink() {  
    resizeBy(0, -50);  
}
```

</Script>

<Body onMouseOver="grow()" onMouseOut="shrink()">

将视窗放大与缩小

</Body>

例5：

<Script>

```
function scrollIt() {  
    for (y=1; y<=2000; y++) {  
        scrollTo(1,y);  
    }  
}
```

</Script>

<Body onDblClick=scrollIt()>

双击鼠标，画面会自动卷动...

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

... The End ...

</Body>

- **open**方法的窗口规格参数：（ yes/no , 1/0 ）

格式 : [var 新窗口对象名  
=]window.open("url","windowName","windowFeature")

alwaysLowered	是否将窗口显示的堆栈后推一层
alwaysRaised	是否将窗口显示的堆栈上推一层
dependent	是否将该窗口与当前窗口产生依存关系
fullscreen	是否满屏显示
directories	是否显示连接工具栏
location	是否显示网址工具栏
menubar	是否显示菜单工具栏
scrollbars	是否显示滚动条
status	是否显示状态栏
titlebar	是否显示标题栏
toolbar	是否显示标准工具栏
resizable	是否可以改变窗口的大小
screenX	窗口左边界距离
screenY	窗口上边界距离
top	窗口上边界
width	窗口宽度
height	窗口高度
left	窗口左边界
outerHeight	窗口外边界的高度
personalbar	是否显示个人工具栏

注释 : open("", "", "menubar")  
           open("", "", "menubar=1")  
           open("", "", "menubar=yes")

例1 :  
       <Script>

```
document.write ("文哥网络技术学习网");
open ('1.htm','','height=100,width=300');
```

```
</Script>
```

```
<!----- 1.htm ----->
<TITLE>欢迎光临</TITLE>

<BODY onClick="self.close()">

<IMG SRC="images\welcome.gif" ALIGN="left">
    <CENTER>
<FONT COLOR="blue" SIZE="5">欢迎光临</FONT><BR>
    这是一个技术研习的小天地<BR>
    </CENTER>

</BODY>
```

例2：

```
<Script>
```

```
document.write("文哥网络技术学习网")
helloWin = open ('1.htm','','height=100,width=300');

var line1 = "<FONT COLOR='blue' SIZE='5'>欢迎光临</FONT>
            <BR>"
var line2 = "这是一个技术研习的小天地"
    helloWin.document.write (line1 + line2)
    helloWin.document.title = "欢迎光临"

</Script>
```

例3：

```
<Script>
```

```
window.defaultStatus = "最棒的线上学习网站";
```

```
</Script>

<A HREF="http://www.hubert.idv.tw" onMouseOver="status='内容
    很充实喔！';return true">
    文哥网路技术学习网</A>
```

例4：  
<Script>

```
queryWin = open('1.htm','height=100,')
```

```
</Script>
```

利用子窗口来改变母窗口的底色

```
/* ----- 1.htm ----- */
<Script>
```

```
function passToOpener(color) {
    opener.document.bgColor = color;
    self.close();
}
```

```
</Script>
<CENTER>
```

请选择你喜欢的颜色?<BR>

```
<FONT onClick="passToOpener('#CCFFFF')"> >浅蓝色
<</FONT>
```

```
<FONT onClick="passToOpener('#FFCCFF')"> >浅红色<</FONT>
```

```
</CENTER>
```



- 屏幕对象：  
描述屏幕的显示及颜色属性。
  - 屏幕对象的属性：
- 格式：screen.属性

availHeight	屏幕区域的可用高度
availWidth	屏幕区域的可用宽度
colorDepth	颜色深度 256/8 16/16 32M/32
height	屏幕区域的实际高度
width	屏幕区域的实际宽度

例1：  
<Script>

```
with (document) {
  write ("您的屏幕显示设定值如下：<P>");
  write ("屏幕的实际高度为", screen.availHeight, "
        <BR>"); 
  write ("屏幕的实际宽度为", screen.availWidth, "
        <BR>"); 
  write ("屏幕的色盘深度为", screen.colorDepth, "
        <BR>"); 
  write ("屏幕区域的高度为", screen.height, "<BR>"); 
  write ("屏幕区域的宽度为", screen.width);
}
```

</Script>

例2：  
<Script>

```
if ( screen.width < 800 || screen.colorDepth < 8 ){
```

```
var msg = "本网站最佳浏览模式为 800 * 600 * 256";  
        alert(msg);  
    }
```

```
</Script>
```

- 事件对象：  
当事件发生时，浏览器自动建立该对象，并包含该事件的类型、鼠标坐标等。
  - 事件对象的属性：
- 格式：event.属性

|         |                                |
|---------|--------------------------------|
| data    | 返回拖拽对象的URL字符串 ( dragDrop )     |
| width   | 该窗口或框架的高度                      |
| height  | 该窗口或框架的高度                      |
| pageX   | 光标相对于该网页的水平位置                  |
| pageY   | 光标相对于该网页的垂直位置                  |
| screenX | 光标相对于该屏幕的水平位置                  |
| screenY | 光标相对于该屏幕的垂直位置                  |
| target  | 该事件被传送到的对象                     |
| type    | 事件的类型                          |
| which   | 数值表示的键盘或鼠标键：1/2/3 ( 左键/中键/右键 ) |
| layerX  | 光标相对于事件发生层的水平位置                |
| layerY  | 光标相对于事件发生层的垂直位置                |
| x       | 相当于layerX                      |
| y       | 相当于layerY                      |

例1：  
 <Script>

```

function getEvent(evnt) {
  eventWin = open ("",'width=200,height=100');
  with (eventWin.document) {
    write("事件类型：" , event.type);
    write("<br>鼠标的x坐标：" , event.screenX);
    write("<br>鼠标的y坐标：" , event.screenY);
  }
}
  
```

```
    }  
    }  
}
```

```
document.write ("单击...")  
document.onmousedown = getEvent;  
</Script>
```

例2：  
<Script>

```
function getCoordinate(evnt) {  
  
    if (document.all) {  
        x = event.screenX;  
        y = event.screenY;  
    }  
    else {  
        x = evnt.screenX;  
        y = evnt.screenY;  
    }  
    status = "水平坐标 :" + x + " ; 垂直坐标 :" + y;  
}  
  
document.onmousemove = getCoordinate;  
  
</Script>
```

例3：  
<Script>

```
function whichKey(evnt) {  
  
    if (document.all) {  
        x = event.button;  
        if( x==1 ) alert("你单击了左键");  
        if( x==2 ) alert("你单击了右键");  
    }  
}
```

```
        else {
            x = evnt.button;
            if( x==1 ) alert("你单击了左键");
            if( x==3 ) alert("你单击了右键");
            return false;
        }
    }

document.onmousedown = whichKey;
document.write("请单击鼠标左/右键");

</Script>
```

- 历史对象：  
用以存储客户端最近访问的网址清单。

格式：

history.属性  
history.方法(参数)

- 历史对象的属性：

|          |               |
|----------|---------------|
| current  | 当前历史记录的网址     |
| length   | 存储在记录清单中的网址数目 |
| next     | 下一个历史记录的网址    |
| previous | 上一个历史记录的网址    |

- 历史对象的方法：

|            |               |
|------------|---------------|
| back()     | 回到上一个历史记录中的网址 |
| forward()  | 回到下一个历史记录中的网址 |
| go(整数或URL) | 前往历史记录中的网址    |

例1：  
`history.go(-1)`

`newWin.history.back()`

`parent.downFrame.histrooy.back()`

例2：

<A HREF="#" onClick="history.back()">上一页</A>

<A HREF="javascript:history.back()">上一页</A>

- 

**位置对象 :**

用来代表特定窗口的URL信息。

**格式 :**

location.属性  
location.方法(参数)

- 

**URL的格式 :**

protocol://host:port/path#hash?search

- 

**URL的种类 :**

|              |                |
|--------------|----------------|
| javascript:  | javascript程序代码 |
| view-source: | 显示源代码          |
| http:        |                |
| file:        |                |
| ftp:         |                |
| mailto:      |                |
| news:        |                |
| gopher       |                |

- 

**位置对象的属性 :**

|          |           |
|----------|-----------|
| hash     | 锚点名称      |
| host     | 主机名称      |
| hostname | host:port |
| href     | 完整的URL字符串 |
| pathname | 路径        |

|          |      |
|----------|------|
| port     | 端口   |
| protocol | 协议   |
| search   | 查询信息 |

- 位置对象的方法：

|             |              |
|-------------|--------------|
| reload()    | 重新加载         |
| replace(网址) | 用指定的网页取代当前网页 |

例1：

按下按钮前往「文哥网络技术学习网」<P>  
<INPUT TYPE="button" VALUE="走吧！" onClick="location.href='http://www.hubert.idv.tw'">

例2：

<FONT COLOR="red"  
onClick="location='http://www.hubert.idv.tw/'">  
文哥网络技术学习网</FONT><P>

<FONT COLOR="blue" STYLE="cursor:hand"  
onClick="location='http://www.hubert.idv.tw/'">  
文哥网络技术学习网</FONT>

例3：

<Script>

```
var sec = 5;

function countDown() {
    if (sec > 0) {
        num.innerHTML = sec--;
    }
    else
        location = "http://www.hubert.idv.tw/";
```

```
        }
    </Script>
```

```
<BODY onLoad="setInterval('countDown()', 1000)"
      <CENTER>
          文哥网络技术学习网
          <H2>http://www.hubert.idv.tw/</H2>
          五秒钟后自动带你前往<BR>
          <FONT ID="num" SIZE="7" FACE="impact">5</FO
```

例4：

```
<A HREF="#" onClick="this.style.behavior='url(#default#homepage)';this.setHomePage('http://www.hubert.idv.tw')
    <FONT COLOR="red"><U>设为首页</U></FONT></A>

<A HREF="javascript:window.external.AddFavorite('http://www.hubert.idv.tw');><FONT COLOR="red"><U>加入收藏</U></FONT></A>
```

- 

### 文件对象：

代表当前HTML对象，是由<body>标签组构成的，对每个HTML文件会自动建立一个文件对象。

### 格式：

document.属性  
document.方法(参数)

- 

### 文件对象的属性：

linkColor	设置超链接的颜色
alinkColor	作用中的超链接的颜色
vlinkColor	链接的超链接颜色
links	以数组索引值表示所有超链接
URL	该文件的网址
anchors	以数组索引值表示所有锚点
bgColor	背景颜色
fgColor	前景颜色
classes	文件中的class属性
cookie	设置cookie
domain	指定服务器的域名
formName	以表单名称表示所有表单
forms	以数组索引值表示所有表单
images	以数组索引值表示所有图像
layers	以数组索引值表示所有layer
embeds	文件中的plug-in
applets	以数组索引值表示所有applet
plugins	以数组索引值表示所有插件程序
referrer	代表当前打开文件的网页的网址
tags	指出HTML标签的样式
title	该文档的标题

width	该文件的宽度 ( px )
lastModified	文件最后修改时间

- 文件对象的方法：

captureEvents(事件)	设置要获取指定的事件
close()	关闭输出字符流，强制显示数据内容
getSelection()	取得当前选取的字串
handleEvent(事件)	使事件处理器生效
open([mimeType, [replace]])	打开字符流
releaseEvents(事件类型)	释放已获取的事件
routeEvent(事件)	传送已捕捉的事件
write(字串)	写字串或数值到文件中
writeln(字串)	分行写字串或数值到文件中 (<pre>.. </pre> )

- 文件对象的事件处理程序：

onClick onDbClick onKeyDown onKeyPress onKeyUp onMouseDown  
onMouseOver

例1：  
<Script>

```
document.bgColor = "white";
document.fgColor = "black";
document.linkColor = "red";
document.alinkColor = "blue";
document.vlinkColor = "purple";
```

</Script>

测试文件对象的颜色属性 : <BR>

<A HREF="http://www.hubert.idv.tw">文哥网路技术学习网</A>

例2：  
<Script>

```
var update_date = document.lastModified;  
var formated_date = update_date.substring(0,10);
```

```
document.write("本网页更新日期：" + update_date + "<BR>")  
document.write("本网页更新日期：" + formated_date)
```

</Script>

- 

锚点对象：

网页中的锚点均会被自动看作锚点对象，并依顺序，分别表示为document.anchors[0]，document.anchors[1]...

定义锚点对象的格式：

字串.anchor(属性)

- 

锚点对象的属性：

name	锚点名称
text	锚点字串

- **链接对象：**  
网页中的链接均会被自动看作链接对象，并依顺序，分别表示为document.links[0]，document.links[1]...

**定义链接对象的格式：**

**字串.link(属性)**

- **链接对象的属性：**

hash	URL中的锚点名称
host	主机域名或IP地址
hostname	URL中的host+port
href	完整的URL字串
pathname	URL中path部分
port	URL中端口部分
protocol	URL中通讯协议部分
search	URL中查询字串部分
target	代表目标的窗口
text	表示A标签中的文字
x	链接对象的左边界
y	链接对象的右边界

- **链接对象的方法：**

handleEvent(事件)  
激活对某事件的处理程序。
- **链接对象的事件处理程序：**

onClick onDbClick onKeyDown onKeyPress onKeyUp  
onMouseDown onMouseUp onMouseOver onMouseOut

例1：  
<Script>

```
function linkGetter() {  
    msgWindow = open","",width=250,height=200)  
msgWindow.document.write("共有" + document.links.length  
+ "个搜索引擎")  
for (var i = 0; i < document.links.length; i++) {  
msgWindow.document.write("<LI>" + document.links[i])  
}  
}  
  
</Script>
```

常用的搜索引擎：<BR>

```
<A HREF="http://www.yam.org.tw/">蕃薯藤</A>  
<A HREF="http://www.kimo.com/">奇摩</A>  
<A HREF="http://chinese.yahoo.com/">雅虎</A>  
<A HREF="http://gais.cs.ccu.edu.tw/">盖世</A>  
<A HREF="http://www.openfind.com.tw/">网擎</A>  
<A HREF="http://www.dreamer.com.tw/">梦想家</A>  
<BR>  
<INPUT TYPE="button" VALUE="网址一览"  
onClick=linkGetter()>
```

- 框架对象：

可以被窗口中的框架引用的对象，具有窗口对象的属性和方法。

格式：

top.frameName|frames[n].属性|方法

parent.frameName|frames[n].属性|方法

例：

```
/* ----- frameset.htm ----- */  
<Script>
```

```
document.title = "框架组页";  
var usrID = "来宾";
```

```
</Script>
```

```
<FRAMESET COLS="20%,80%">  
<FRAME SRC="menu.htm" NAME=leftFrame>  
    <FRAMESET ROWS="10%,90%">  
        <FRAME SRC="usrInfo.htm" NAME=upFrame>  
            <FRAME SRC="welcome.htm"  
                NAME=downFrame>  
        </FRAMESET>  
    </FRAMESET>
```

```
<!----- menu.htm ----->
```

```
<HEAD>  
<TITLE>导航页</TITLE>  
<STYLE>A{text-decoration:none}</STYLE>
```

```
</HEAD>
<BODY>
<CENTER>
<A HREF="login.htm" TARGET="downFrame">会员登
录</A><BR>
<A HREF="hot.htm" TARGET="downFrame">热门优惠
</A><BR>
<A HREF="welcome.htm" TARGET="downFrame">回
首页</A>
<BR>
</CENTER>
</BODY>
```

```
/* ----- userInfo.htm ----- */
<Script>
```

```
document.title = "用户信息";
var bye = "欢迎有空常来..."
```

```
document.write
("<MARQUEE>亲爱的<FONT COLOR='gray'>
<U>",top.usrID,"</U></FONT>会员，欢迎您的光临！
</MARQUEE>");
```

```
</Script>
```

```
/* ----- login.htm ----- */
<Script>
```

```
document.title = "用户登录";
```

```
function login() {
top.usrID = document.loginForm.usr.value;
top.upFrame.location = "userInfo.htm";
```

```
    }

</Script>

<HTML>
<FORM NAME="loginForm">
    请输入您的大名：
    <INPUT TYPE="text" NAME="usr">
    <INPUT TYPE="button" VALUE="填写完毕"
        onClick=login()
    </FORM>
</HTML>
```

- 防止直接链接：

例：

```
<Script> /* ----- hot.htm ----- */

document.title = "热门优惠";

if (top.usrID == null) {
    location = "frameset.htm";
}
```

```
</Script>
```

今日优惠：

```
<LI>超级豪华大比萨。 ( 原价$550 , 今天只要$450 )
<LI>海鲜大比萨。 ( 原价$550 , 今天只要$450 )
```

- 检查是否使用框架：

例：

```
<!----- welcome.htm ----->
```

```
<HEAD>
<TITLE>首页</TITLE>
<STYLE>A{text-decoration:none}</STYLE>
```

```
<Script>

if (top.frames.length == 0) {
    location = "frameset.htm";
}
if (top.frames.length > 0) {
    location = "frameset.htm";
}

</Script>

</HEAD>
<CENTER>
欢迎光临<H1>
<FONT COLOR="green" FACE="arial">
    My PIZZA</FONT></H1>
</CENTER>
```

- 

### 表单对象：

文件对象的子对象，Javascript的runtime engine自动为每一个表单建立一个表单对象。

#### 格式：

document.forms[索引].属性

document.forms[索引].方法（参数）

document.表单名称.属性

document.表单名称.方法（参数）

- 

### 表单对象的属性：

action	表单动作
elements	以索引表示的所有表单元素
encoding	MIME的类型
length	表单元素的个数
method	方法
name	表单名称
target	目标

- 

### 表单对象的方法：

handleEvent(事件)	使事件处理程序生效
reset()	重置
submit()	提交

1.

### 文本对象：

◦

格式：

document.forms[索引].elements[索引].属性

document.forms[索引].elements[索引].方法(参数)

document.表单名称.文本对象名称.属性

document.表单名称.文本对象名称.方法(参数)

◦

属性：

defaultValue	该对象的value属性
form	该对象所在的表单
name	该对象的name属性
type	该对象的type属性
value	该对象的value属性

◦

方法：

blur()	
focus()	
handleEvent(事件)	
select()	该对象设置为选取状态

◦

事件处理程序：

onBlur onChange onClick onDbClick onFocus onKeyDown  
onKeyPress onKeyUp  
onMouseDown onMouseUp onMouseOver onMouseOut  
onMouseMove onSelect

◦

例1：

```
<FORM>
姓名：
<INPUT TYPE="text" NAME="name">
```

```
<INPUT TYPE="button" VALUE="请单击"
onClick=alert("谢谢你 , "+this.form.name.value)>
</FORM>
```

- 例2：

```
<Script>
```

```
function getFocus(obj) {
    obj.style.color='red';
    obj.style.background='#DBDBDB';
}
function getBlur(obj) {
    obj.style.color='black';
    obj.style.background='white';
}
```

```
</Script>
```

```
<BODY onLoad=document.form1.name.focus()
    <FORM NAME="form1">
姓    名 : <INPUT TYPE="text" NAME="name"
onFocus=getFocus(this) onBlur=getBlur(this)><BR>
电    话 : <INPUT TYPE="text" NAME="tel"
onFocus=getFocus(this) onBlur=getBlur(this)><BR>
    </FORM>
</BODY>
```

- 例3：

```
<Script>
```

```
var i = 0;

function movenext(obj,i) {
    if(obj.value.length==4){
        document.forms[0].elements[i+1].focus();
    }
}
```

```

        function result() {
            fm = document.forms[0];
            num = fm.elements[0].value +
                fm.elements[1].value +
                fm.elements[2].value +
                fm.elements[3].value ;
            alert("你输入的信用卡号码是"+ num);
        }

    </Script>

<BODY onLoad=document.forms[0].elements[i].focus()>
    请输入你的信用卡号码 :
    <form>
        <input type=text size=3 maxlength=4
            onkeyup=movenext(this,0)> -
        <input type=text size=3 maxlength=4
            onkeyup=movenext(this,1)> -
        <input type=text size=3 maxlength=4
            onkeyup=movenext(this,2)> -
        <input type=text size=3 maxlength=4
            onkeyup=movenext(this,3)>
        <input type=button value=显示 onClick=result()>
    </form>
</BODY>

```

2.  
◦

密码对象：  
格式：

document.forms[索引].elements[索引].属性

document.forms[索引].elements[索引].方法(参数)

document.表单名称.密码对象名称.属性

document.表单名称.密码对象名称.方法(参数)

◦

属性：

defaultValue	该对象的value属性
--------------	-------------

form	该对象所在的表单
name	该对象的name属性
type	该对象的type属性
value	该对象的value属性

◦ 方法：

blur()	
focus()	
handleEvent(事件)	
select()	该对象设置为选取状态

◦ 事件处理程序：

onBlur onChange onClick onDbClick onFocus onKeyDown  
 onKeyPress onKeyUp  
 onMouseDown onMouseUp onMouseOver onMouseOut  
 onMouseMove onSelect

◦ 例1：

<Script>

```

        function checkPw() {
            fm = document.form1;
            if (fm.pw2.value != fm.pw1.value) {
                alert("密码不符，请重新输入");
                document.form1.pw2.select();
            }
            else
                alert("谢谢你， "+fm.name.value);
        }
    
```

</Script>

<BODY onLoad=document.form1.name.focus()>

<FORM NAME="form1">

姓 名：<INPUT TYPE="text" NAME="name"><BR>

```
输入密码 : <INPUT TYPE="password" NAME="pw1">
             <BR>
重新输入 : <INPUT TYPE="password" NAME="pw2">
             <BR>
             <INPUT TYPE="button" VALUE="填好了"
                    onClick=checkPw()
             <INPUT TYPE="reset" VALUE="重 填">
             </FORM>
             </BODY>
```

◦

例2：

```
<Script>
```

```
function isInt(elm) {
    if (isNaN(elm)) {
        alert("你输入的是" + elm + "\n不是数字！");
        document.forms[0].pw.value = "";
        return false;
    }
    if (elm.length != 4) {
        alert("请输入四位数数字！");
        document.forms[0].pw.value = "";
        return false;
    }
}
</Script>
```

```
<form action="test.asp" onSubmit="return
    isInt(this.pw.value)">
    请输入四位数数字密码 : <BR>
    <input type="password" name="pw">
    <input type="submit" value="检查">
</form>
```

3. 按钮对象、提交按钮对象、重置按钮对象：

◦

格式：

document.forms[索引].elements[索引].属性

document.forms[索引].elements[索引].方法(参数)

document.表单名称.按钮对象名称.属性

document.表单名称.按钮对象名称.方法(参数)

◦

属性：

form	该对象所在的表单
name	该对象的name属性
type	该对象的type属性
value	该对象的value属性

◦

方法：

blur()	
click()	
focus()	
handleEvent(事件)	

◦

事件处理程序：

onBlur onClick onDbClick onFocus onKeyDown onKeyPress  
onKeyUp

onMouseDown onMouseUp onMouseOver onMouseOut  
onMouseMove

4.

隐藏对象：

◦

格式：

document.forms[索引].elements[索引].属性

`document.forms[索引].elements[索引].方法(参数)`

`document.表单名称.隐藏对象名称.属性`

`document.表单名称.隐藏对象名称.方法(参数)`

◦ 属性：

<code>form</code>	该对象所在的表单
<code>name</code>	该对象的name属性
<code>type</code>	该对象的type属性
<code>value</code>	该对象的value属性

5. 单选按钮对象：

◦ 格式：

`document.forms[索引].elements[索引](索引).属性`

`document.forms[索引].elements[索引](索引).方法(参数)`

`document.表单名称.单选对象名称[索引].属性`

`document.表单名称.单选对象名称[索引].方法(参数)`

◦ 属性：

<code>checked</code>	设置该对象为选定状态
<code>defaultChecked</code>	对应该对象的默认选定状态
<code>form</code>	该对象所在的表单
<code>name</code>	该对象的name属性
<code>type</code>	该对象的type属性
<code>value</code>	该对象的value属性

◦ 方法：

blur()	
click()	
focus()	
handleEvent(事件)	

- 事件处理程序：

onBlur onClick onDbClick onFocus onKeyDown onKeyPress  
 onKeyUp  
 onMouseDown onMouseUp onMouseOver onMouseOut  
 onMouseMove

- 例：

```
<Script>

    function show() {
        var x = "先生";
        if (document.form1.sex[1].checked)
            x = "小姐";
        alert(document.form1.name.value + x);
    }

</Script>
<FORM NAME=form1>
姓名 : <INPUT TYPE="text" NAME="name"><BR>
你是 : <INPUT TYPE="radio" NAME="sex" CHECKED>
    帅哥
    <INPUT TYPE="radio" NAME="sex">美女<BR>
    <INPUT TYPE="button" VALUE="请单击"
        onClick=show()
    </FORM>
```

## 6. 复选框对象：

- 格式：

document.forms[索引].elements[索引].属性

document.forms[索引].elements[索引].方法(参数)

document.表单名称.单选对象名称[索引].属性

document.表单名称.单选对象名称[索引].方法(参数)

◦ 属性：

checked	设置该对象为选定状态
defaultChecked	对应该对象的默认选定状态
form	该对象所在的表单
name	该对象的name属性
type	该对象的type属性
value	该对象的value属性

◦ 方法：

blur()	
click()	
focus()	
handleEvent(事件)	

◦ 事件处理程序：

onBlur onClick onDbClick onFocus onKeyDown onKeyPress  
onKeyUp  
onMouseDown onMouseUp onMouseOver onMouseOut  
onMouseMove

◦ 例：

<Script>

```
function count() {  
    var checkCount=0;  
    var num = document.form1.elements.length;
```

```

        for (var i=0; i<num; i++) {
            if (document.form1.elements[i].checked)
                checkCount++;
        }
        alert ("你喜欢 "+ checkCount + "种颜色。 ")
    }

</Script>
<FORM NAME=form1>
请选择你喜欢的颜色 : <BR>
<INPUT TYPE="checkbox" NAME="red">红色
<INPUT TYPE="checkbox" NAME="green">绿色
<INPUT TYPE="checkbox" NAME="blue">蓝色<BR>
<INPUT TYPE="button" VALUE="请单击"
       onClick=count()>
</FORM>

```

## 7. 选择对象 :

- 属性 :

|               |            |
|---------------|------------|
| form          | 该对象所在的表单   |
| name          | 该对象的name属性 |
| length        | 选项的数目      |
| options       | <option>标记 |
| selectedIndex | 所选项目的索引值   |
| type          | 该对象的type属性 |

- 方法 :

|                 |  |
|-----------------|--|
| blur()          |  |
| focus()         |  |
| handleEvent(事件) |  |

- 事件处理程序 :

onBlur onClick onChange onFocus onKeyDown onKeyPress  
onKeyUp  
onMouseDown onMouseUp onMouseOver onMouseOut  
onMouseMove

## 8. 选项对象：选择对象的子对象

- 格式：

<option value="值" selected>文字</option>

new Option([文字[,值[,defaultSelected[,selected]]]])

- 属性：

|                 |               |
|-----------------|---------------|
| selected        | 判断该选项是否被选取    |
| defaultSelected | 指定该选项为默认选定状态  |
| index           | 所有选项所构成的数组索引值 |
| length          | 选项的数目         |
| text            | 该选项显示的文字      |
| value           | 所选项传到服务器的值    |

- 例1：

<Script>

```
var url = new Array(3);
url[0] = "http://www.yam.org.tw/";
url[1] = "http://www.kimo.com/";
url[2] = "http://chinese.yahoo.com/";
```

```
function goto(form) {
var i = form.menu.selectedIndex;
if (i>0) {
location = url[i-1];
}
}
```

```
</Script>
<FORM>
<SELECT NAME="menu" onChange="goto(this.form)">
    <OPTION>你喜欢哪一个搜索引擎?
    <OPTION STYLE="color:red">◆蕃薯藤
    <OPTION STYLE="color:red">◇奇摩
    <OPTION STYLE="color:red">◆中文雅虎
</SELECT>
</FORM>
```

◦ 例2：

```
<Script>

    function getText() {
        sel = document.forms[0].weekday
        ans = sel.options[sel.selectedIndex].text
        return ans;
    }

</Script>
<FORM>请选择...
<SELECT name=weekday>
<OPTION VALUE="Monday">星期一
<OPTION VALUE="Tuesday">星期二
<OPTION VALUE="Wednesday">星期三
<OPTION VALUE="Thursday">星期四
    <OPTION VALUE="Friday">星期五
<OPTION VALUE="Saturday">星期六
    <OPTION VALUE="Sunday">星期日
</SELECT><P>
<INPUT TYPE="button" VALUE="取出选项的文字"
        onClick="alert(getText())">
<INPUT TYPE="button" VALUE="取出选项的值"
        onClick="alert(this.form.weekday.value)"><BR>
</FORM>
```

◦ 例3：

```
<FORM NAME="form1">
    你最喜欢哪一种水果?
    <INPUT TYPE="text" NAME="fruit">
        <A HREF="#" 
onClick="javascript:open('1.htm','','width=100')">
            查询</A>
        </FORM>
```

```
<Script> /* ----- 1.htm ----- */
```

```
    function choice() {
sel = document.forms[0].elements[0];
        document.form1.fruit.value =
sel.options[sel.selectedIndex].text;
        self.close();
    }
```

```
</Script>
```

```
<FORM>
<SELECT onChange="choice()">
    <OPTION>请选择
    <OPTION>西瓜
    <OPTION>香蕉
    </SELECT>
</FORM>
```

◦ 例4：

```
<Script>
```

```
    function createOptions(){
var option = new Option(document.form1.select1.value)
        document.form1.select2.options[2] = option;
    }
```

```
</script>
```

```
<form name="form1">
  <select name="select1" size="10">
    <option>可选择项目 <option>-----
    <option value="香蕉">香蕉 <option value="葡萄">葡萄
    <option value="苹果">苹果 <option value="梨子">梨子
  </select>

  <input type="button" value="-->" onClick="createOptions()">
  <select name="select2" size="10">
    <option>选择项目 <option>-----
  </select>
</form>
```

◦ 例5：

```
<Script>

  function createOptions(){
    sel1 = document.form1.select1;
    sel2 = document.form1.select2;
    var num = sel1.selectedIndex;
    var option = new Option(sel1.options[num].text);
    sel2.options[2] = option;
  }

</script>

<form name="form1">
  <select name="select1" size="10">
    <option>可选择项目 <option>-----
    <option value="香蕉">香蕉 <option value="葡萄">葡萄
    <option value="苹果">苹果 <option value="梨子">梨子
  </select>

  <input type="button" value="-->" onClick="createOptions()">
  <select name="select2" size="10">
    <option>选择项目 <option>-----
```

```
</select>
</form>
```

- 例6：

```
<Script>
```

```
    function createOptions(){
        sel1 = document.form1.select1;
        sel2 = document.form1.select2;
        var num = sel1.selectedIndex;
        var option = new Option(sel1.options[num].text);
        var item = sel2.options.length;
        sel2.options[item] = option;
    }
```

```
</script>
```

```
<form name="form1">

    <select name="select1" size="10">
        <option>可选择项目 <option>-----
        <option value="香蕉">香蕉 <option value="葡萄">葡萄
        <option value="苹果">苹果 <option value="梨子">梨子
    </select>

    <input type="button" value="-->" 
        onClick="createOptions()"
        <select name="select2" size="10">
            <option>选择项目 <option>-----
        </select>
    </form>
```

- 例7：

```
<Script>
```

```
function createOptions(){
    sel1 = document.form1.select1;
```

```

        sel2 = document.form1.select2;
        var num = sel1.selectedIndex;
        if (num > 1) {
            var option = new Option(sel1.options[num].text);
            var item = sel2.options.length;
            sel2.options[item] = option;
        }
        sel1.selectedIndex = 10000;
    }

    function delOptions() {
        var num = document.form1.select2.selectedIndex;
        if (num>1)
            document.form1.select2.options[num] = null;
        else
            document.form1.select2.selectedIndex = 10000;
    }

</script>
<form name="form1">
<select name="select1" size="10"
    onClick="createOptions()">
<option>可选择项目 <option>-----
<option value="香蕉">香蕉 <option value="葡萄">葡萄
<option value="苹果">苹果 <option value="梨子">梨子
</select>
<input type="button" value="选择"
    onClick="createOptions()">
<select name="select2" size="10">
<option>选择项目 <option>-----
</select>
<input type="button" value="删除" onClick="delOptions()">
</form>

```

9.

文本区域对象：

- 

属性：

defaultValue	对应该对象的默认值
form	该对象所在的表单
name	该对象的name属性
type	该对象的type属性
value	该对象的value属性

- 方法：

blur()	
select()	
focus()	
handleEvent(事件)	

- 事件处理程序：

onBlur onClick onChange onSelect onFocus onKeyDown  
                   onKeyPress onKeyUp  
 onMouseDown onMouseUp onMouseOver onMouseOut  
                   onMouseMove

- 例1：

<Script>

```

function isTooLong(elm){
  if (elm.length > 50) {
    alert("留言内容太长，请修改后再发送....");
    return false;
  }
}

```

</script>

```

<FORM onSubmit="return isTooLong(this.msg.value)">
<TEXTAREA NAME="msg" COLS="30" ROWS="5"
          onFocus="this.value=''"
          欢迎留言，不过请长话短说....
          </textarea><BR>

```

```
<INPUT TYPE="submit" VALUE="留言完毕">
</FORM>
```

- 例2：

```
<STYLE>
INPUT {background-color:99FFFF;color:"red"}
TEXTAREA {background-color:99FFFF;color:"red"}
</STYLE>
<BODY BGCOLOR="99FFFF">
<FORM METHOD="post" ENCTYPE="text/plain"
ACTION="mailto:hwyang@iii.org.tw?subject=不错">
<TABLE>
<CAPTION>读者回函</CAPTION>
<TR><TD>姓 名：
<TD><INPUT TYPE="text" NAME="userName">
<TR><TD>电子邮件：
<TD><INPUT TYPE="text" NAME="email">
<TR><TD VALIGN="top">内 容：
<TD><TEXTAREA NAME="msg" ROWS="2"
COLS="30">
我非常喜欢你的书，加油!!!
</TEXTAREA>
<TR><TD COLSPAN="2" ALIGN="center">
<INPUT TYPE="submit" VALUE="填好了">
</TABLE>
</FORM>
</BODY>
```

## 10.

### 文件上传对象：

- 属性：

form	该对象所在的表单
name	该对象的name属性
type	该对象的type属性
value	该对象的value属性

◦

方法：

blur()	
select()	
focus()	
handleEvent(事件)	

◦

事件处理程序：

onBlur onClick onSelect onFocus onKeyDown onKeyPress  
onKeyUp  
onMouseDown onMouseUp onMouseOver onMouseOut  
onMouseMove