



## Tarena High-End IT Training

加拿大达内科技(中国)公司

## JavaScript网页开发技术

[www.tarena.com](http://www.tarena.com)

### 中国北京:

电 话: 62196102, 62196104  
地 址: 海淀区北三环西路18号  
中鼎大厦B座701、712.  
Email: [hansy@tarena.com](mailto:hansy@tarena.com)

### 加拿大多伦多:

Tel : 1-(647) 284-8872  
Address: 106-7 Crescent  
Place Toronto,  
Ontario, Canada  
Post Code: M4C 5L7

CONFIDENTIAL

1 外企的师资、外企的技术、外企的品质

[www.Tarena.com](http://www.Tarena.com)

## 课程目标

达内科技

- u 掌握JavaScript的语法、程序控制结构
- u 熟练使用JavaScript内置基本对象
- u 熟悉JavaScript调用DHTML元素
- u 掌握用JavaScript编写客户端脚本程序的技能
- u 能创建功能强大的互动网页

北京: 010-62196102 上海: 61202630

2

外企的师资、外企的技术、外企的品质

- JavaScript概述
- JavaScript基础语法
- JavaScript常用内置对象
- JavaScript常用DHTML对象
- JavaScript高级技巧

- JavaScript概述
- JavaScript基础语法
- JavaScript常用内置对象
- JavaScript常用DHTML对象
- JavaScript高级技巧

- JavaScript是一种网页编程技术，大部分使用者将它用于创建动态交互网页
- JavaScript是一种基于对象和事件驱动的解释性脚本语言，具有与Java和C语言类似的语法
- JavaScript是一种使用简单，功能强大的编程语言

本练习主要针对事件驱动和基于对象，该例子在网页的一个按钮中设置了onclick属性，当鼠标单击该按钮时，将会弹出一个窗口，浏览google搜索网站

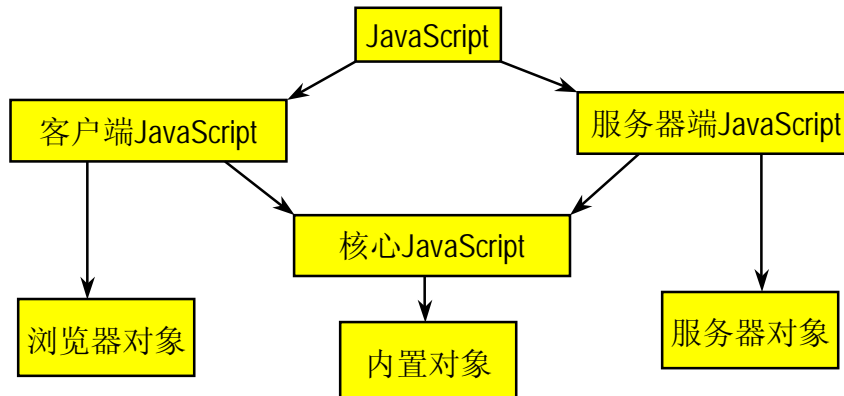
- window是客户端JavaScript中浏览器对象之一，open是window对象的一个方法，意思是打开窗口，在参数中设置好网址，就可以打开指定的网站。在将来学习DHTML的时候将会详细讲解window对象的使用方法

打开记事本，将以下代码输入，保存为D:\hello.htm

```
<html><body>
<input type="button"
  onclick="window.open('http://www.google.com')">
</body></html>
```

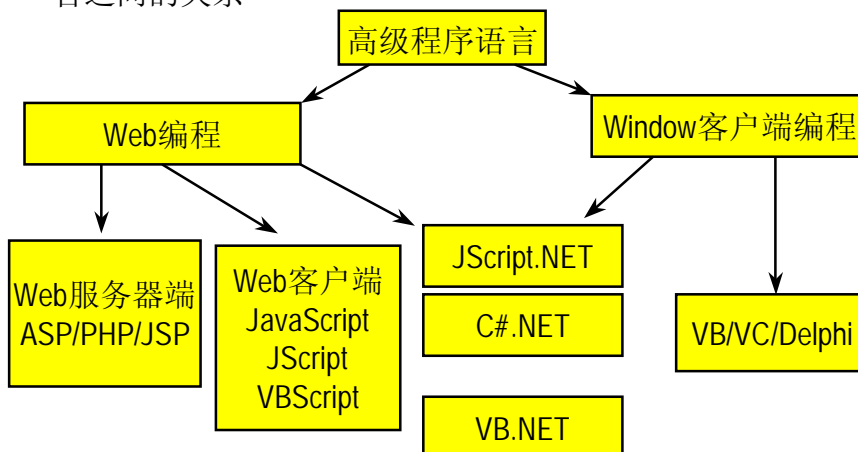
保存好以后双击该文件打开，试一试单击按钮后运行的结果

JavaScript的基础为核心JavaScript，在实现上分为客户端JavaScript、服务器端JavaScript



北京: 010-62196102 上海: 61202630 7 外企的师资、外企的技术、外企的品质

JavaScript与ASP(JScript)、JScript.Net及JSP等编程语言之间的关系



北京: 010-62196102 上海: 61202630 8 外企的师资、外企的技术、外企的品质

- ▣ 面向对象的敲门砖
- ▣ 具备完整的语法
- ▣ 随着Web浏览器的改进，功能越来越强大
- ▣ 是搭配服务器端技术的主要客户端编程语言

- ▣ 网景公司在Netscape2.0首先推出了JavaScript
- ▣ 微软公司从IE3.0开始提供对客户端JavaScript的支持，并另取名为JScript
- ▣ 微软公司从IIS3.0开始提供对服务器端JScript的支持
- ▣ 微软推出JScript.NET，基于.NET框架的JScript具备了和C#.NET和VB.NET一样强大的功能

### ┌ 简单易用

- ┆ 简洁易用，与Java有类似的语法
- ┆ 可以使用任何文本编辑工具编写
- ┆ 只需要浏览器就可以执行程序

### ┌ 解释执行

- ┆ 事先不编译
- ┆ 逐行执行
- ┆ 无需进行严格的变量声明

### ┌ 基于对象

- ┆ 内置大量现成对象，编写少量程序可以完成目标

### ┌ 适合做哪些事情

- ┆ 客户端数据计算
- ┆ 客户端表单合法性验证
- ┆ 浏览器对象的调用
- ┆ 浏览器事件的触发
- ┆ 网页特殊显示效果制作

### ┌ 不适合做哪些事情

- ┆ 大型应用程序
- ┆ 图像、多媒体处理
- ┆ 网络实时通讯应用

- u JavaScript与Java运行方式不一样
  - | JavaScript是解释执行
  - | Java是编译执行
- u JavaScript不是Java的简化版本
  - | 逻辑运算符、语句结构类似
  - | 变量申明、对象调用等不同
- u JavaScript和Java功能实现不一样
  - | JavaScript通过浏览器实现程序功能
  - | Java通过Java虚拟机实现程序功能

	独立性	运行方式	变量声明	特长
Java Applet	可独立运行	需编译后由JVM执行	需要明确定义	网络通讯、图形、多媒体操作
JavaScript	嵌入HTML运行	无需编译, 由浏览器执行	不需要明确定义	浏览器对象操作

- JavaScript的编辑工具很多，如Microsoft FrontPage、DreamWeaver、Microsoft Visual InterDev等，都可以用来进行JavaScript网页制作
- 针对简单的程序，我们使用文本编辑器直接书写源代码

- 在定义事件时直接写入JavaScript脚本

```
<input type="button" value="打开google网站"
onclick="window.open('http://www.google.com')">
```



## 运行结果



用记事本输入程序，以文件名 test1\_1.htm保存，用IE打开该文件



鼠标单击按钮  
[打开google网站]后弹出新  
的google站点IE窗口



## 在网页中直接嵌入JavaScript

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

```
<!--
/*
```

HTML注释

脚本开始声明

```
程序功能：书写方法说明
开始和结束的标志的书写方法
单行和多行JavaScript注释的写法
*/
```

JavaScript多行注释

```
function sayhello(){
    //在HTML文档中显示hello
    document.write("hello");
}
```

JavaScript单行注释

语句结尾

```
sayhello();
```

脚本结束声明

```
-->
```

```
</script>
```

## 运行结果

用记事本输入程序，  
以文件名test1-2.htm保存，  
用IE打开该文件



```
<script language="JavaScript">
<!--
// 单行注释，老脚本注释和
开始和结束标记的书写方法，单行和块级JavaScript程序的写法
//
function sayhello(){
    // 在HTML文档中显示hello
    document.write("hello");
}
sayhello();
-->
</script>
```

IE浏览器执行JavaScript  
时忽略了JS注释部分  
，只输出运行结果

## 在网页中调用独立JavaScript文件

```
<script language="JavaScript" src="test1-3.js"></script>
```

HTML文件

```
<!--
function sayhello(){
    //在HTML文档中显示hello
    document.write("hello from js");
}
sayhello();
-->
```

JS脚本文件中不  
需要脚本开始和  
结束声明

### 运行结果

用记事本分别创建test1\_3.htm和test1-3.js，用IE打开test1\_3.htm文件



文件调入JavaScript代码和直接嵌入一样可以正常运行

- 在Windows、Linux、Unix操作系统都可以运行JavaScript，只要安装了支持JavaScript的浏览器
- 不同的浏览器甚至同一浏览器的不同版本对JavaScript的支持程度都不一样
- 大部分常见的浏览器都对JavaScript提供支持，在本教程中以IE6.0作为主要的调试运行环境

## 支持JavaScript的浏览器与操作系统

浏览器	Windows	Linux	Unix
IE 3.0--6.0	√	×	√
Netscape 3.0-7.0	√	√	√
Mozilla 1.0--1.6	√	√	√
Opera 5.0—7.0	√	√	√
Firebird	×	√	√
Konqueror	×	√	×
Galeon	×	√	×

支持：√ 不支持：×

北京：010-62196102 上海：61202630

23

外企的师资、外企的技术、外企的品质

## IE的升级

### 为什么要升级IE

- 更加完善与稳定
- 对ECMA v3国际标准的支持更加完整

### 如何升级IE

- 下载安装

[http://www.microsoft.com/windows/ie\\_intl/cn/](http://www.microsoft.com/windows/ie_intl/cn/)

北京：010-62196102 上海：61202630

24

外企的师资、外企的技术、外企的品质

## u Mozilla的安装

### ┆ 为什么要安装Mozilla

- u 完全符合ECMA v3国际标准
- u Netscape6采用与其一致的内核
- u 支持Windows、Linux和Unix等更多操作系统
- u 源代码开放、程序短小、装卸方便
- u 可用于检查JavaScript在非IE运行情况

### ┆ 如何安装Mozilla

- u 下载Mozilla  
<http://www.mozilla.org/>
- u 运行Mozilla安装程序

## u 如果程序出现错误

- ┆ 双击左下角出现的感叹号
- ┆ 弹出错误显示窗口，详细错误所在的行号



- JavaScript概述
- JavaScript基础语法
- JavaScript常用内置对象
- JavaScript常用DHTML对象
- JavaScript高级技巧

JavaScript程序是由Unicode字符集编写的，每个字符和汉字都是采用2个字节进行编码

- ASCII码由大小写英文字母、数字、英文符号等组成，采用1个字节中的低7位进行编码，是Unicode编码的子集
- Unicode是一种国际编码，可以表达几乎任何书写语言，它是采用16位编码的字符集

ASCII编码：7位编码

0 1 0 0 0 0 0 1



A

Unicode编码：16位编码

1 0 0 1 1 1 1 0

1 1 0 0 0 0 0 0



你

在HTML中大小写是不敏感的，但是在JavaScript程序中大小写是敏感的

- HTML都是大小写不敏感的
- 标准的JavaScript语法定义中是区分大小写

```
<input type=button name="HelloBtn" value="play" onClick="myclick()">
<script language="JavaScript">
function myclick(){
    HelloBtn.value="pause"
}
</script>
```

大小写须一致

换行、分号

有换行，分号  
允许不加

推荐加上分号减少  
错误和歧义的发生

a=1;b=2;

=

a=1  
b=2

=

a=1;  
b=2;

空格、TAB

提倡加上空格或TAB  
增强程序可读性

```
{
a=1;
b=2;
}
```

=

```
{
    a=1;
    b=2;
}
```

## └ 常量

- ┆ 直接在程序中出现的数据值

## └ 标识符

- ┆ 由不以数字开头的字母、数字、下划线(\_)、美元符号(\$)组成
- ┆ 常用于表示函数、变量等的名称
  - ┆ 例如: `_abc`, `$abc`, `abc`, `abc123` 是标识符, 而 `1abc` 不是
- ┆ JavaScript语言中代表特定含义的词称为保留字, 不允许程序再定义为标识符

## └ ECMA v3标准保留的JavaScript的关键字

break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with		



### 使用范例

SayHello是自定义标识符

function、var是保留字

字符串常量

```
<script language="JavaScript">
<!--
function SayHello(){
    var hellostr="您好";
    document.write(hellostr);
}
SayHello();
-->
</script>
```

### 词法结构

### 数据类型

#### 数据类型分类

#### 数字类型

#### 字符串类型

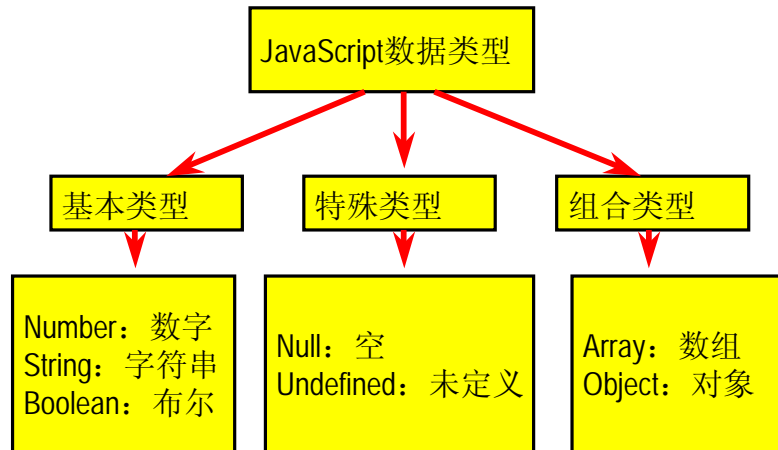
#### 类型转换

### 运算符

### 语句

### 变量与函数

### 对象



### 简介

- 最基本的数据类型
- 不区分整型数值和浮点型数值
- 所有数字都采用64位浮点格式存储，相当于Java和C语言中的double格式
- 能表示的最大值是 $\pm 1.7976931348623157 \times 10^{308}$
- 能表示的最小值是 $\pm 5 \times 10^{-324}$

## └ 整数

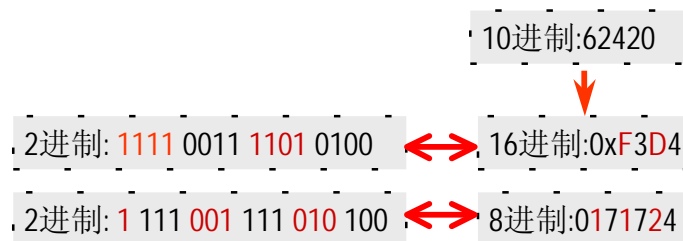
- 在JavaScript中10进制的整数由数字的序列组成
- 精确表达的范围是  
-9007199254740992 ( $-2^{53}$ ) 到 9007199254740992 ( $2^{53}$ )
- 超出范围的整数，精确度将受影响

## └ 浮点数

- 使用小数点记录数据
  - 例如: 3.4, 5.6
- 使用指数记录数据
  - 例如:  $4.3e23 = 4.3 \times 10^{23}$

## └ 16进制和8进制数的表达

- 16进制数据前面加上0x，八进制前面加0
- 16进制数是由0-9,A-F等16个字符组成
- 8进制数由0-7等8个数字组成
- 16进制和8进制与2进制的换算



## 简介

- 是由Unicode字符、数字、标点符号组成的序列
- 字符串常量首尾由单引号或双引号括起
- JavaScript中没有字符类型
- 常用特殊字符在字符串中的表达
  - 字符串中部分特殊字符必须加上右划线\
  - 常用的转义字符

换行	单引号	双引号	右划线
\n	\'	\"	\\

## String数据类型的使用

- 特殊字符的使用方法和效果
- Unicode的插入方法

```
//测试特殊字符的书写  
var aa="\u4f60\u597d\n欢迎来到JavaScript世界";  
alert(aa);
```

你

好

换行



### 简介

- Boolean类型仅有两个值：true和false，也代表1和0，实际运算中true=1,false=0
- 布尔值也可以看作on/off、yes/no、1/0对应true/false
- Boolean值主要用于JavaScript的控制语句，例如：

```
if (x==1){  
    y=y+1;  
}else{  
    y=y-1;  
}
```

### 简介

- Null在程序中代表变量没有值或者不是一个对象
- Undefined代表变量的值尚未指定或者对象属性根本不存在

## 有趣的比较

比较	结果
null与空字符串	不相等，null代表什么也没有，空字符串则代表一个为空的字符串
null与false	不相等，但是!null等于true
null与0	不相等，但是在C++等其他语言中是相等的
null与undefined	相等，但是null与undefined并不相同

## JavaScript属于松散类型的程序语言

- 变量在声明的时候并不需要指定数据类型
- 变量只有在赋值的时候才会确定数据类型
- 表达式中包含不同类型数据则在计算过程中会强制进行类别转换

数字 + 字符串: 数字转换为字符串

数字 + 布尔值: true转换为1, false转换为0

字符串 + 布尔值: 布尔值转换为字符串true或false

└ 强制类型转换函数

- ┆ 函数parseInt: 强制转换成整数

- ┆ 例如parseInt("6.12")=6

- ┆ 函数parseFloat: 强制转换成浮点数

- ┆ 例如parseFloat("6.12")=6.12

- ┆ 函数eval: 将字符串强制转换为表达式并返回结果

- ┆ 例如eval("1+1")=2,eval("1<2")=true

└ 类型查询函数

- ┆ 函数typeof: 查询数值当前类型  
(string / number / boolean / object )

- ┆ 例如

- typeof("test")+3)="string",typeof(null)="object"

└ 加(+)、减(-)、乘(\*)、除(/)、余数(%)

└ 加、减、乘、除、余数和数学中的运算方法一样

└ 例如:  $9/2=4.5$ ,  $4*5=20$ ,  $9\%2=1$

└ -除了可以表示减号还可以表示负号

└ 例如:  $x=-y$

└ +除了可以表示加法运算还可以用于字符串的连接

└ 例如: `"abc"+"def"="abcdef"`

└ 递增(++)、递减(--)

└ 假如 $x=2$ , 那么 $x++$ 表达式执行后的值为3,  $x--$ 表达式执行后的值为1

└  $i++$ 相当于 $i=i+1$ ,  $i--$ 相当于 $i=i-1$

└ 递增和递减可以放在变量前也可以放在变量后



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

与 ( && ) 、 或 ( || ) 、 非 ( ! )

1 && 1 = 1	1    1 = 1	! 1 = 0
1 && 0 = 0	1    0 = 1	! 0 = 1
0 && 0 = 0	0    0 = 0	

左移 ( << ) 、 右移 ( >> ) 、 NOT ( ~ )

$3 \ll 2 = 12$

A horizontal row of four squares representing the binary value 3. The first two squares are white with '0', and the last two are red with '1'.

A horizontal row of four squares representing the binary value 12. The first two squares are red with '1', and the last two are white with '0'. An orange arrow points from the third square of the top row to the first square of this row.

左移2位后补0

$12 \gg 1 = 6$

A horizontal row of four squares representing the binary value 12. The first two squares are red with '1', and the last two are white with '0'.

A horizontal row of four squares representing the binary value 6. The first square is white with '0', and the next three are red with '1', '1', '0'. An orange arrow points from the second square of the top row to the second square of this row.

右移1位前补0

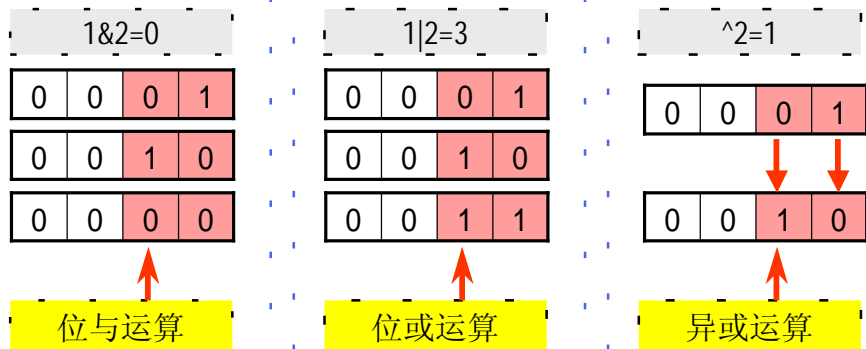
$\sim 6 = 9$

A horizontal row of four squares representing the binary value 6. The first square is white with '0', and the next three are red with '1', '1', '0'. A red arrow points from the first square of this row to the first square of the bottom row.

A horizontal row of four squares representing the binary value 9. The first and last squares are red with '1', and the middle two are white with '0', '0'. A red arrow points from the first square of the top row to the first square of this row.

位非1和0互换

└ 位与(&)、位或(|)、异或(^)



└ 赋值 =

- JavaScript中=代表赋值，两个等号==表示相等
- 例如， $x=1$ 表示给x赋值为1
- if ( $x==1$ ){...}程序表示当x与1相等时

└ 配合其他运算符形成的简化表达式oP=

- 例如 $i+=1$ 相当于 $i=i+1$ ， $x\&y$ 相当于 $x=x\&y$

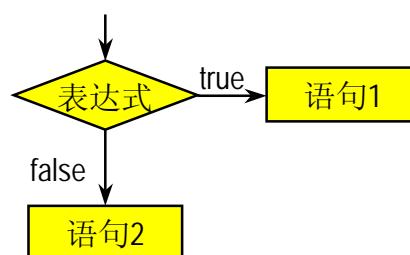
## u 优先顺序

- 按照右表从上到下的优先顺序执行

运算符	描述
( )	括号
++, --, -, ~, !	一元运算符
*, /, %	乘法、除法、取模
+, -, +	加法、减法、字符串连接
<<, >>, >>>	移位
<, <=, >, >=	小于、小于等于、大于、大于等于
==, !=, ===, !==	等于、不等于、恒等、不恒等
&	按位与
^	按位异或
	按位或
&&	逻辑与
	逻辑或
?:	条件运算符
=, oP=	赋值、运算赋值

## u if-else基本格式

```
if (表达式){  
    语句 1;  
    .....  
}else{  
    语句 2;  
    .....  
}
```



## u 功能说明

- 如果表达式的值为true则执行语句1，否则执行语句2

## 程序范例

```
var x= (new Date()).getDay();  
//获取今天的星期值, 0为星期天  
var y;
```

```
if ((x==6) || (x==0)) {  
    y="周末";  
}else{  
    y="工作日";  
}
```

等价于

if 语句允许不使用  
else子句

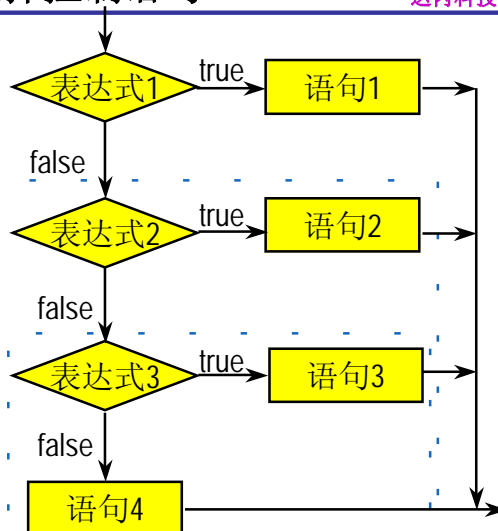
```
y="工作日";  
if ((x==6) || (x==0)) {  
    y="周末";  
}
```

## if 语句嵌套格式

```
if (表达式1) {  
    语句1;  
}else if (表达式2){  
    语句2;  
}else if (表达式3){  
    语句3;  
} else{  
    语句4;  
}
```

## 功能说明

执行顺序参见右图



## 程序范例

- 对变量x的值进行判断，采用if语句嵌套转换成相应的星期名称

if 语句允许进行嵌套

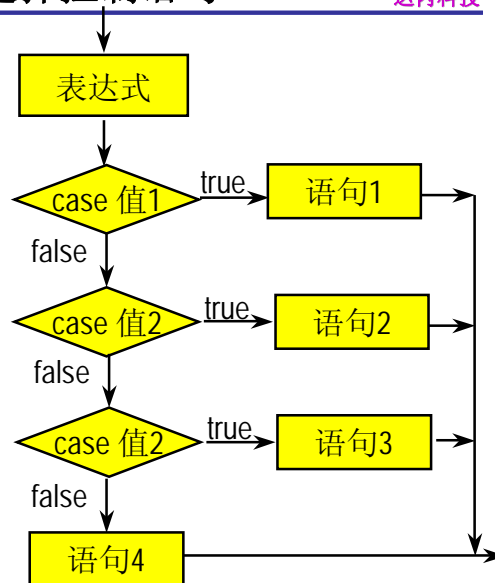
```
if (x==1){  
    y="星期一";  
}else if (x==2){  
    y="星期二";  
...  
}else if (x==6){  
    y="星期六";  
}else if (x==0){  
    y="星期日";  
}else{  
    y="未定义";  
}
```

## switch基本格式

```
switch (表达式) {  
    case 值1:语句1;break;  
    case 值2:语句2;break;  
    case 值3:语句3;break;  
    default:语句4;  
}
```

## 功能说明

- 详细的计算过程参考右图



## 程序范例

- 对变量x的值进行判断，采用switch转换成相应的星期名称

switch比else if结构更加简洁清晰

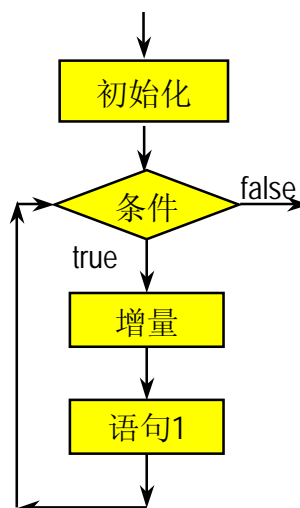
```
switch(x){  
case 1:y="星期一";break;  
case 2:y="星期二";break;  
case 3:y="星期三";break;  
case 4:y="星期四";break;  
case 5:y="星期五";break;  
case 6:y="星期六";break;  
case 7:y="星期日";break;  
default: y="未定义";  
}
```

## for循环基本格式

```
for (初始化;条件;增量){  
    语句1;  
    ...  
}
```

## 功能说明

- 实现条件循环，当条件成立时，执行语句1，否则跳出循环体



### 程序范例

```
for (var i=1;i<=7;i++){  
    document.write("<H"+i+">hello</H "+i+"> ");  
    document.write("<br>");  
}  
//循环输出H1到H7的字体大小
```

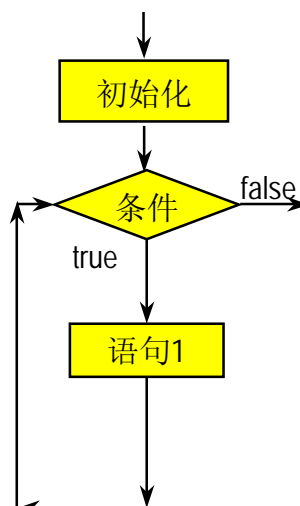


### while循环基本格式

```
while (条件){  
    语句1;  
    ...  
}
```

### 功能说明

- 运行功能和for类似，当条件成立循环执行语句花括号{}内的语句，否则跳出循环



## 程序范例

```
var i=1;
while (i<=7) {
    document.write("<H"+i+">hello</H "+i+"> ");
    document.write("<br>");
    i++;
}
//循环输出H1到H7的字体大小
```



采用while和for的输出效果一样

## 变量申明语句

```
var a;
```

不需要指定变量类型

```
var x,y;
```

允许一次定义多个变量

## 变量赋值语句

```
a=1;
```

赋值为数值

```
x="hello,how are you!";
```

赋值为字符串

```
y=x;
```

赋值为另一个变量

## 变量的调用

```
...
```

```
alert(a);
```

```
document.write("Tom,"+x);
```

```
...
```



## u 函数的定义

```
function 函数名 (参数){  
    函数体;  
    return 返回值;  
}
```

## u 功能说明

- | 可以使用变量、常量或表达式作为函数调用的参数
- | 函数由关键字function定义
- | 函数名的定义规则与标识符一致，大小写是敏感的
- | 返回值必须使用return

## u 使用范例

```
<script language="JavaScript">  
/* Sayhello是定义的函数名，前面必须加上function和空格*/  
function SayHello(){  
    var hellostr;  
    var myname=prompt("请问您贵姓? ","陈");  
    hellostr="您好, "+myname+'先生, 欢迎进入"探索之旅"! ';  
    alert(hellostr);  
    document.write(hellostr);  
}  
//这里是对前面定义的函数进行调用  
SayHello();  
</script>
```

在调用函数的时候要注意函数的大小写，如果写成sayhello或sayHello都会出错

### 对象的分类:

- | 对象由属性和方法封装而成
- | JavaScript是一种基于对象语言，对象是JavaScript中最重要元素
- | JavaScript包含四种对象
  - ü 内置对象
  - ü 自定义对象
  - ü 浏览器对象
  - ü ActiveX对象

### 11种内置对象

- | 包括
  - ü **Array ,String , Date, Math, Boolean, Number**
  - ü **Function, Global, Error, RegExp , Object**
- | 简介
  - ü 在JavaScript中除了null和undefined以外其他的数据类型都被定义成了对象
  - ü 也可以用创建对象的方法定义变量
  - ü String、Math、Array、Date、RegExp都是JavaScript中重要的内置对象

在JavaScript程序大多数功能都是通过对象实现

```
<script language="javascript">
var aa=Number.MAX_VALUE;
//利用数字对象获取可表示最大数
var bb=new String("hello JavaScript");
//创建字符串对象
var cc=new Date();
//创建日期对象
var dd=new Array("星期一","星期二","星期三","星期四");
//数组对象
</script>
```

- JavaScript概述
- JavaScript基础语法
- JavaScript常用内置对象
- JavaScript常用DHTML对象
- JavaScript高级技巧

## 11种内置对象

## 包括

- Array, Boolean, Date, Math, Number, String
- Error, Function, Global, Object, RegExp

## 简介

- 在JavaScript中除了null和undefined以外其他的数据类型都被定义成了对象
- 可以用创建对象的方法定义变量
- String、Math、Array、Date是JavaScript中常用的对象

类型	内置对象	介绍
数据对象	Number	数字对象
	String	字符串对象
	Boolean	布尔值对象
组合对象	Array	数组对象
	Math	数学对象
	Date	日期对象
高级对象	Object	自定义对象
	Error	错误对象
	Function	函数对象
	RegExp	正则表达式对象
	Global	全局对象

### └ 自动创建字符串对象

```
var str1="hello world";  
alert(str1.length);  
alert(str1.substr(1,5));
```

调用字符串的对象属性或方法时自动创建对象，用完就丢弃

### └ 手工创建字符串对象

```
var str1= new String("hello word");  
alert(str1.length);  
alert(str1.substr(1,3));
```

采用new创建字符串对象str1，全局有效

### └ 获取字符串长度

┆ length

### 书写格式

- x.length

### 使用注解

- x代表字符串对象
- length必须小写
- 中间用点操作符调用
- 汉字、字母长度均为1
- 返回大于或等于0整数

```
var str1="String对象";  
var str2="";  
alert("str1长度 "+str1.length);  
alert("str2长度 "+str2.length);
```



### String对象的方法分类



方法	介绍	方法	介绍
anchor()	锚	fontcolor(color)	字体颜色
blink()	闪烁		
fixed()	固定		
bold()	粗体	fontsize(size)	字体大小
italics	斜体		
strike()	删除线		
big()	字变大	link(url)	超链接
small()	字变小		
sub()	下标		
sup()	上标		

## u 书写格式

- string对象提供了一组针对HTML格式的方法，如x.anchor()返回锚定义字符串<a>x</a>，x.bold()返回粗体表示字符串<b>x</b>，x.sup()返回上标格式字符串<sup>x</sup>

## u 举例说明

```
var x="google";
var y="x.bold():"+x.bold();
document.write(y.fontsize(7));
//相当于document.write("<font size=7><b>google</b></font>")
```



格式编排方法返回值列表

方法	返回值	说明
anchor()	<a>string</a>	返回锚定义字符串
blink()	<blink>string</bin	返回闪烁定义字符串
fixed()	<tt>string</tt>	返回固定定义字符串
bold()	<b>string</b>	返回粗体定义字符串
italics()	<i>string</i>	返回斜体定义字符串
strike()	<strike>string</st	返回删除线定义字符串
big()	<big>string</big>	返回字变大定义字符串
small()	<small>string</sma	返回字变小定义字符串
sub()	<sub>string</sub>	返回下标定义字符串
sup()	<sup>string</sup>	上标定义字符串

北京: 010-62196102 上海: 61202630 79 外企的师资、外企的技术、外企的品质

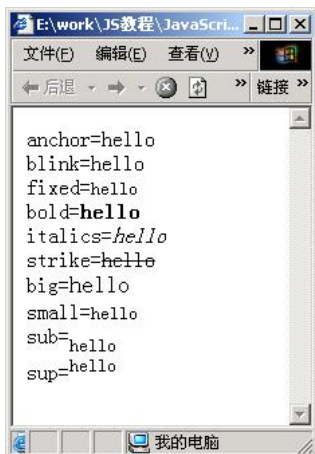
格式编排方法返回值列表

方法	返回值	说明
fontcolor(color)	<font color="color">string	返回字体颜色定义字符串, color参数可以为red、blue、green等
fontsize(size)	</font><font size="size">string	返回字体大小定义字符串, size从小到大可以定义为1到7
link(url)	</font><a href="url">string</a>	返回超链接定义字符串, url为网络超链接

北京: 010-62196102 上海: 61202630 80 外企的师资、外企的技术、外企的品质



## 使用范例



```
function writeln(mystr){
    document.write(mystr+"<br>");
}
var x="hello";
writeln("anchor="+x.anchor());
writeln("blink="+x.blink());
writeln("fixed="+x.fixed());
writeln("bold="+x.bold());
writeln("italics="+x.italics());
writeln("strike="+x.strike());
writeln("big="+x.big());
writeln("small="+x.small());
writeln("sub="+x.sub());
writeln("sup="+x.sup());
```

类别	方法	说明
大小写转换	toLowerCase()	返回小写字符串
	toUpperCase()	返回大写字符串
获取指定字符	charAt(index)	返回指定位置字符
	charCodeAt(index)	返回指定位置字符Unicode编码
查询字符串	indexOf(findstr, index)	返回正向的索引位置
	lastIndexOf(findstr)	返回反向的索引位置
	match(regex)	返回匹配的字符串
	search(regex)	返回找到字符串的首字符索引

## String对象的方法(2) -- 大小写转换... 达内科技

### 书写格式

- | x.toLowerCase()
- | x.toUpperCase()

### 使用注解

- | x代表字符串对象
- | 返回转换后的字符串

```
var str1="AbcdEfgh";
```

```
var str2=str1.toLowerCase();  
var str3=str1.toUpperCase();  
alert(str2);  
//结果为"abcdefgh"  
alert(str3);  
//结果为"ABCDEFGH"
```



北京: 010-62196102 上海: 61202630

83

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 大小写转换 达内科技

### 程序图解

```
var str1="AbcdEfgh";
```

A	b	c	d	E	f	g	h
---	---	---	---	---	---	---	---

```
var str2=str1.toLowerCase();
```

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

```
var str3=str1.toUpperCase();
```

A	B	C	D	E	F	G	H
---	---	---	---	---	---	---	---

```
alert(str2);  
//结果为"abcdefgh"
```

```
alert(str3);  
//结果为"ABCDEFGH"
```

北京: 010-62196102 上海: 61202630

84

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 获取指定字符...达内科技

### u 书写格式

- x.charAt(index)
- x.charCodeAt(index)

### u 使用注解

- x代表字符串对象
- index代表字符位置
- index从0开始编号
- charAt返回index位置的字符
- charCodeAt返回index位置的Unicode编码

```
var str1="JavaScript网页教程";
var str2=str1.charAt(12);
var str3=str1.charCodeAt(12);
alert(str2);
//结果为"教"
alert(str3);
//结果为25945
```



北京: 010-62196102 上海: 61202630

85

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 获取指定字符达内科技

### u 程序图解

```
var str1="JavaScript网页教程";
```

J	a	v	a	S	c	r	i	p	t	网	页	教	程
0	1	2	3	4	5	6	7	8	9	10	11	12	13

```
var str2=str1.charAt(12);
```

教

```
var str3=str1.charCodeAt(12);
```

25945

```
alert(str2);
//结果为"教"
alert(str3);
//结果为25945
```

北京: 010-62196102 上海: 61202630

86

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 查询字符串... 达内科技

### u 书写格式

- | x.indexOf(findstr,index)
- | x.lastIndexOf(findstr)

### u 使用注解

- | x代表字符串对象
- | findstr代表查找的字符串
- | index代表开始找的位置索引，省略代表从开始找
- | 返回findstr在x中出现的首字符位置索引，如果没有找到返回-1
- | lastIndexOf代表从后面找起

```
var str1="JavaScript网页教程";  
var str2=str1.indexOf("a");  
var str3=str1.lastIndexOf("a");  
alert(str2);  
//结果为1  
alert(str3);  
//结果为3
```



北京: 010-62196102 上海: 61202630

87

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 查询字符串... 达内科技

### u 程序图解

```
var str1="JavaScript网页教程";
```

J	a	v	a	S	c	r	i	p	t	网	页	教	程
0	1	2	3	4	5	6	7	8	9	10	11	12	13

```
var str2=str1.indexOf("a");
```

```
var str3=str1.lastIndexOf("a");
```

```
alert(str2);
```

```
//结果为1
```

```
alert(str3);
```

```
//结果为3
```

北京: 010-62196102 上海: 61202630

88

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 查询字符串...达内科技

### u 书写格式

- x.match(regex)
- x.search(regex)

### u 使用注解

- x代表字符串对象
- regex代表正则表达式或字符串
- match返回匹配字符串的数组，如果没有匹配则返回null
- search返回匹配字符串的首字符位置索引

```
var str1="JavaScript网页教程";

var str2=str1.match("Script");
var str3=str1.search("Script");
alert(str2[0]);
//结果为"Script"
alert(str3);
//结果为4
```



北京: 010-62196102 上海: 61202630

89

外企的师资、外企的技术、外企的品质

## String对象的方法(2) -- 查询字符串 达内科技

### u 程序图解

```
var str1="JavaScript网页教程";
```

J	a	v	a	S	c	r	i	p	t	网	页	教	程
0	1	2	3	4	5	6	7	8	9	10	11	12	13

match返回的是  
字符串数组

search的作用同  
indexOf方法

```
var str2=str1.match("Script");
var str3=str1.search("Script");
```

```
alert(str2[0]);
//结果为"Script"
alert(str3);
//结果为4
```

北京: 010-62196102 上海: 61202630

90

外企的师资、外企的技术、外企的品质

类别	方法	说明
获取子字符串	substr(start, length)	返回从索引位置start开始长为length的子字符
	substring(start, end)	返回start开始end结束的子字符串
	slice(start, end)	同substring, 但允许使用负数表示从后计算位
替换子字符串	replace(findstr, tostr)	返回替换findstr为tostr之后的字符串
分割字符串	split(bystr)	返回由bystr分割成的字符串数组
连接字符串	concat(string)	返回与string连接后的字符串

#### u 书写格式

- | x.substr(start, length)
- | x.substring(start, end)

#### u 使用注解

- | x代表字符串对象
- | start表示开始位置
- | length表示截取长度
- | end是结束位置加1
- | 第一个字符位置为0

```
var str1="abcdefgh";
var str2=str1.substr(2,4);
var str3=str1.substring(2,4);
alert(str2);
//结果为"cdefg"
alert(str3);
//结果为"cd"
```



## String对象的方法(3) -- 获取子字符串

### 程序图解

```
var str1="abcdefgh";
```

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

```
var str2=str1.substr(2,4);
```

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

从索引为2的字符开始，  
截取四个字符

```
alert(str2);  
//结果为"cdef"
```

```
var str3=str1.substring(2,4);
```

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

从索引为2的字符开始，  
到索引为3的字符为止

```
alert(str3);  
//结果为"cd"
```

北京: 010-62196102 上海: 61202630

93

外企的师资、外企的技术、外企的品质

## String对象的方法(3) -- 获取子字符串

### 书写格式

- x.slice(start, end)

### 使用注解

- x代表字符串对象
- start表示开始位置索引
- end是结束位置下一字符索引编号
- 第一个字符索引为0
- start、end可为负数  
-1代表最后一个字符
- end省略则相当于从start位置开始截取以后所有字符

```
var str1="abcdefgh";  
var str2=str1.slice(2,4);  
var str3=str1.slice(4);  
var str4=str1.slice(2,-1);  
var str5=str1.slice(-3,-1);  
alert(str2);  
//结果为"cd"  
alert(str3);  
//结果为"efgh"  
alert(str4);  
//结果为"cdefg"  
alert(str5);  
//结果为"fg"
```

北京: 010-62196102 上海: 61202630

94

外企的师资、外企的技术、外企的品质

## String对象的方法(3) -- 获取子字符串

### 程序图解

```
var str2=str1.slice(2,4);  
var str3=str1.slice(4);  
var str4=str1.slice(2,-1);  
var str5=str1.slice(-3,-1);  
alert(str2);  
//结果为"cd"  
alert(str3);  
//结果为"efgh"  
alert(str4);  
//结果为"cdefg"  
alert(str5);  
//结果为"fg"
```

```
var str1="abcdefgh";
```

a	b	c	d	e	f	g	h
0	1	2	3	4	5	6	7
-	-	-	-	-4	-	-	-
8	7	6	5	4	3	2	1

a	b	c	d	e	f	g	h
0	1	2	3	4	5	6	7
-	-	-	-	-4	-	-	-
8	7	6	5	4	3	2	1

北京: 010-62196102 上海: 61202630

95

外企的师资、外企的技术、外企的品质

## String对象的方法(3) -- 替换子字符串...

### 书写格式

1. x.replace(findstr,tostr)

### 使用注解

- 1. x代表字符串对象
- 1. findstr要找的子字符串
- 1. tostr替换为的字符串
- 1. 返回替换后的字符串

```
var str1="abcdefgh";
```

```
var str2=str1.replace("cd","aaa");
```

```
alert(str2);
```

```
//结果为"abaaefgh"
```



北京: 010-62196102 上海: 61202630

96

外企的师资、外企的技术、外企的品质



## String对象的方法(3) -- 替换子字符串 达内科技

### 程序图解

```
var str1="abcdefgh";
```

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

```
var str2=str1.replace("cd","aaa");
```

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

a	b	a	a	a	e	f	g	h
---	---	---	---	---	---	---	---	---

```
alert(str2);  
//结果为"abaaefgh"
```

将str1字符串中的子字符串"cd"替换为"aaa"

北京: 010-62196102 上海: 61202630

97

外企的师资、外企的技术、外企的品质

## String对象的方法(3) -- 分割字符串... 达内科技

### 书写格式

- x.split(bystr)

### 使用注解

- x代表字符串对象
- bystr作为分割字符串
- 返回分割后的字符串数组

```
var str1="一,二,三,四,五,六,日";
```

```
var strArray=str1.split(",");
```

```
alert(strArray[1]);  
//结果为"二"
```



北京: 010-62196102 上海: 61202630

98

外企的师资、外企的技术、外企的品质

## 程序图解

```
var str1="一,二,三,四,五,六,日";
```

一	,	二	,	三	,	四	,	五	,	六	,	日
---	---	---	---	---	---	---	---	---	---	---	---	---

```
var strArray=str1.split(",");
```

一	,	二	,	三	,	四	,	五	,	六	,	日
---	---	---	---	---	---	---	---	---	---	---	---	---



生成的数组元素编号是从0开始编号的，所以索引1指的是第二个元素

```
alert(strArray[1]);  
//结果为"二"
```

## 书写格式

- y=x.concat(addstr)

## 使用注解

- x代表字符串对象
- addstr为添加字符串
- 返回x+addstr字符串

```
var str1="abcd";  
var str2=str1.concat("efgh");
```

```
alert(str2);  
//结果为"abcdefgh"
```



## 程序图解

```
var str1="abcd";
```

a	b	c	d
---	---	---	---

将str1与字符串"efgh"相加后赋值给str2，但是str1自身的值不发生变化

```
var str2=str1.concat("efgh");
```

a	b	c	d
---	---	---	---

 + 

e	f	g	h
---	---	---	---

```
alert(str2);  
//结果为"abcdefgh"
```

## 创建数组对象

```
var cnweek=new Array(7);  
var books=new Array();
```

创建数组时允许指定元素个数也可以不指定元素个数

## 初始化数组对象

```
var cnweek=new Array(7);  
cnweek[0]="星期日";  
cnweek[1]="星期一";  
...  
cnweek[6]="星期六";
```

```
var test=new Array(100,"a",true);
```

也可以直接在建立对象时初始化数组元素，元素类型允许不同

## └ 创建二维数组

```
var cnweek=new Array(7);  
for (var i=0;i<=6;i++){  
    cnweek[i]=new Array(2);  
}  
cnweek[0][0]="星期日";  
cnweek[0][1]="Sunday";  
cnweek[1][0]="星期一";  
cnweek[1][1]="Monday";  
...  
cnweek[6][0]="星期六";  
cnweek[6][1]="Saturday";
```

通过指定数组中的元素为数组的方式可以创建二维甚至多维数组

星期日	Sunday
星期一	Monday
星期二	Tuesday
星期三	Wednesday
星期四	Thursday
星期五	Friday
星期六	Saturday

## └ 获取数组元素的个数

┆ length

### 书写格式

┆ x.length

### 使用注解

┆ x代表数组对象

┆ length必须小写

┆ 中间用点操作符调用

┆ 返回大于或等于0整数

```
var cnweek=new Array(7);
cnweek[0]="星期日";
cnweek[1]="星期一";
cnweek[2]="星期二";
cnweek[3]="星期三";
cnweek[4]="星期四";
cnweek[5]="星期五";
cnweek[6]="星期六";
for (var i=0;i<cnweek.length;i++){
    document.write(cnweek[i]+" | ");
}
```



类别	方法	说明
连接数组	join(bystr)	返回由bystr连接数组元素组成的字符串
	toString()	返回由逗号(,)连接数组元素组成的字符串
	concat(value,...)	返回添加参数中元素后的数组
数组排序	reverse()	返回反向的数组
	sort()	返回排序后的数组

## u 书写格式

- | x.join(bystr)
- | x.toString()

## u 使用注解

- | x代表数组对象
- | bystr作为连接数组中元素的字符串
- | 返回连接后的字符串
- | 与字符串的split功能刚好相反

```
var arr1=[1, 2, 3,4,5,6,7];
```

```
var str1=arr1.join("-");
```

```
alert(str1);  
//结果为"1-2-3-4-5-6-7"
```



## u 程序图解

```
var arr1=[1, 2, 3,4,5,6,7];
```

1	,	2	,	3	,	4	,	5	,	6	,	7
---	---	---	---	---	---	---	---	---	---	---	---	---

```
var str1=arr1.join("-");
```

1	-	2	-	3	-	4	-	5	-	6	-	7
---	---	---	---	---	---	---	---	---	---	---	---	---

根据join方法传入参数"-"  
将元素连接成字符串

```
alert(str1);  
//结果为"1-2-3-4-5-6-7"
```

## └ 书写格式

└ x.concat(value,...)

## └ 使用注解

└ x代表数组对象

└ value作为数组元素  
连接到数组的末尾

└ 返回连接后的数组

└ concat方法并不改变  
x自身的值

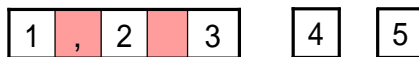
```
var a = [1,2,3];  
var b=a.concat(4, 5) ;
```

```
alert(a.toString());  
//返回结果为1,2,3  
alert(b.toString());  
//返回结果为1,2,3,4,5
```

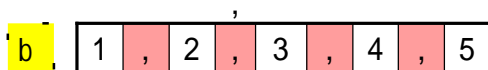
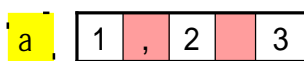


## └ 程序图解

```
var a = [1,2,3];
```



```
var b=a.concat(4, 5) ;
```



```
alert(a.toString());  
//返回结果为1,2,3  
alert(b.toString());  
//返回结果为1,2,3,4,5
```

## u 书写格式

- | x.reverse()
- | x.sort()

## u 使用注解

- | x代表数组对象
- | 返回排序后的数组
- | 排序后x会发生改变

```
var arr1=[32, 12, 111, 444];
```

```
arr1.reverse(); //颠倒数组元素
alert(arr1.toString());
```

```
//结果为444,111,12,32
```

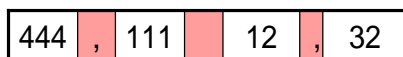
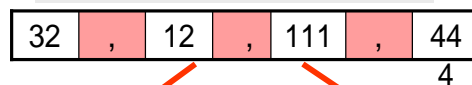
```
arr1.sort(); //排序数组元素
alert(arr1.toString());
```

```
//结果为111,12,32,444
```

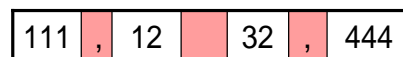


## u 程序图解

```
var arr1=[32, 12, 111, 444];
```



```
arr1.reverse(); //颠倒数组元素
alert(arr1.toString());
//结果为444,111,12,32
```



```
arr1.sort(); //排序数组元素
alert(arr1.toString());
//结果为111,12,32,444
```



类别	方法	说明
获取子数组	slice(start,end)	通过数组元素起始和结束索引号获取子数组
	splice(start, deleteCount, value, ...)	对数组指定位置进行删除和插入
进出栈操作	push(value, ...)	数组末端入栈操作
	pop()	数组末端出栈操作
	unshift(value, ...)	数组首端入栈操作
	shift()	数组首端出栈操作

### u 书写格式

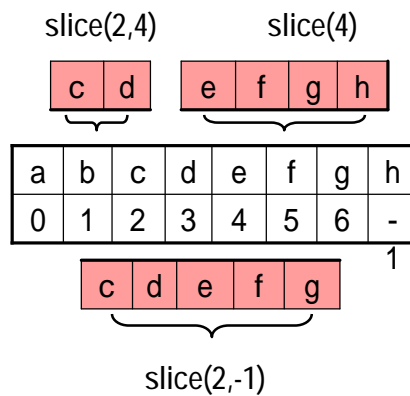
- l x.slice(start, end)

### u 使用注解

- l x代表数组对象
- l start表示开始位置索引
- l end是结束位置下一数组元素索引编号
- l 第一个数组元素索引为0
- l start、end可为负数，-1代表最后一个数组元素
- l end省略则相当于从start位置截取以后所有数组元素

## 程序图解

```
var arr1=['a','b','c','d','e','f','g','h'];  
  
var arr2=arr1.slice(2,4);  
var arr3=arr1.slice(4);  
var arr4=arr1.slice(2,-1);  
  
alert(arr2.toString());  
//结果为"c,d"  
alert(arr3.toString());  
//结果为"e,f,g,h"  
alert(arr4.toString());  
//结果为"c,d,e,f,g"
```



## 书写格式

- ┆ x.splice(start, deleteCount, value, ...)

## 使用注解

- ┆ x代表数组对象
- ┆ splice的主要用途是对数组指定位置进行删除和插入
- ┆ start表示开始位置索引
- ┆ deleteCount删除数组元素的个数
- ┆ value表示在删除位置插入的数组元素
- ┆ value参数可以省略

## 程序图解

```
var a = [1,2,3,4,5,6,7,8]
a.splice(1,2);
//a变为 [1,4,5,6,7,8]
alert(a.toString());
a.splice(1,1);
//a变为[1,5,6,7,8]
alert(a.toString());
a.splice(1,0,2,3);
//a变为[1,2,3,5,6,7,8]
alert(a.toString());
```

splice(1,2)

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	-

splice(1,1)

1	4	5	6	7	8
0	1	2	3	4	5

1	5	6	7	8
0	2	3	4	5

splice(1,0,2,3)

1	2	3	5	6	7	8
0	1	2	4	5	6	-

## 书写格式

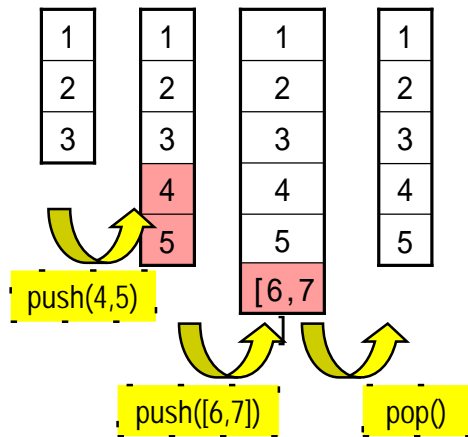
- | x.push(value, ...)
- | x.pop()

## 使用注解

- | x代表数组对象
- | value可以为字符串、数字、数组等任何值
- | push是将value值添加到数组x的结尾
- | pop是将数组x的最后一个元素删除

## 程序图解

```
var arr1=[1,2,3];  
arr1.push(4,5);  
alert(arr1);  
//结果为"1,2,3,4,5"  
arr1.push([6,7]);  
alert(arr1);  
//结果为"1,2,3,4,5,6,7"  
  
arr1.pop();  
alert(arr1);  
//结果为"1,2,3,4,5"
```



## 书写格式

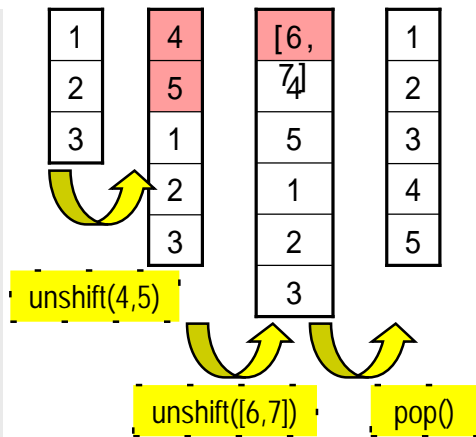
- | x.unshift(value,...)
- | x.shift()

## 使用注解

- | x代表数组对象
- | value可以为字符串、数字、数组等任何值
- | unshift是将value值插入到数组x的开始
- | shift是将数组x的第一个元素删除

## 程序图解

```
var arr1=[1,2,3];  
arr1.unshift(4,5);  
alert(arr1);  
//结果为"4,5,1,2,3"  
arr1.unshift([6,7]);  
alert(arr1);  
//结果为"6,7,4,5,1,2,3"  
  
arr1.shift();  
alert(arr1);  
//结果为"4,5,1,2,3"
```



## Math对象的属性都是数学常数

- Math.E (自然数)
- Math.LN2 ( $\ln 2$ )
- Math.LN10 ( $\ln 10$ )
- Math.LOG2E ( $\log_2 e$ )
- Math.LOG10E ( $\log e$ )
- Math.PI (圆周率)
- Math.SQRT1\_2 (根号 $1/2$ )
- Math.SQRT2 (根号 $2$ )

## u 用程序获取Math属性的值

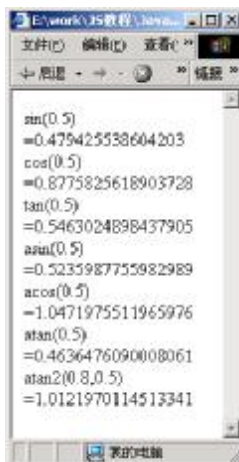
```
//定义换行输出函数
function writeln(mystr){
    document.write(mystr+"<br>");
}
//输出Math属性
writeln("E="+Math.E);
writeln("LN2="+ Math.LN2);
writeln("LN10="+ Math.LN10 );
writeln("LOG2E="+ Math.LOG2E );
writeln("LOG10E="+ Math.LOG10E );
writeln("PI="+ Math.PI );
writeln("SQRT1_2="+ Math.SQRT1_2 );
writeln("SQRT2="+ Math.SQRT2 );
```



类别	方法	说明
三角函数	Math.sin(x)	计算正弦值 (x在0~2 $\pi$ 之间, 返回值-1~1)
	Math.cos(x)	计算余弦值 (x在0~2 $\pi$ 之间, 返回值-1~1)
	Math.tan(x)	计算正切值 (x在0~2 $\pi$ 之间, 返回正切值)

类别	方法	说明
反三角函数	Math.asin(x)	计算反正弦值 (x在 -1与1之间, 返回0~2 $\pi$ )
	Math.acos(x)	计算反余弦值 (x在 -1与1之间, 返回0~2 $\pi$ )
	Math.atan(x)	计算反正切值 (x可以为任意值, 返回 - $\pi/2$ ~ $\pi/2$ )
	Math.atan2(y, x)	计算从X轴到一个点的角度 (y,x分别为点的纵坐标和横坐标, 返回 - $\pi$ ~ $\pi$ )

## 程序示范



```

function writeln(mystr){
    document.write(mystr+"<br>");
}
var x=0.5;
var y=0.8;
writeln("sin("+x+")="+Math.sin(x));
writeln("cos("+x+")="+ Math.cos(x));
writeln("tan("+x+")="+ Math.tan(x));
writeln("asin("+x+")="+ Math.asin(x));
writeln("acos("+x+")="+ Math.acos(x));
writeln("atan("+x+")="+ Math.atan(x));
writeln("atan2("+y+", "+x+")="+ Math.atan2(y,x));

```

类别	方法	说明
计算函数	Math.sqrt(x)	计算平方根
	pow(x,y)	计算 $x^y$
	Math.exp(x)	计算e的指数 $e^x$
	Math.log(x)	计算自然对数 (x为大于0的整数)

## 程序示范

```
function writeln(mystr){  
    document.write(mystr+"<br>");  
}  
var x=3;  
var y=4;  
writeln("sqrt("+x+")="+Math.sqrt(x));  
writeln("pow("+x+","+y+")="+ Math.pow(x,y));  
writeln("exp (" +x+","+y+")="+ Math.exp(x,y));  
writeln("log("+x+")="+ Math.log(x));
```

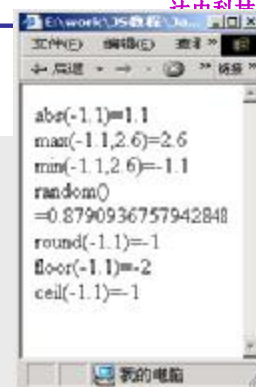




类别	方法	说明
数值比较函数	Math.abs(x)	计算绝对值
	Math.max(x,y)	返回参数中较大的一个
	Math.min(x,y)	返回参数中较小的一个
	Math.random()	计算0~1之间的一个随机数
	Math.round(x)	舍入为最接近的整数
	Math.floor(x)	对一个数下舍入
	Math.ceil(x)	对一个数上舍入

## 程序示范

```
function writeIn(mystr){  
    document.write(mystr+"<br>");  
}  
var x=-1.1;  
var y=2.6;  
writeIn("abs("+x+")="+Math.abs(x));  
writeIn("max("+x+", "+y+")="+ Math.max(x,y));  
writeIn("min("+x+", "+y+")="+ Math.min(x,y));  
writeIn("random()="+ Math.random());  
writeIn("round("+x+")="+ Math.round(x));  
writeIn("floor("+x+")="+ Math.floor(x));  
writeIn("ceil("+x+")="+ Math.ceil(x));
```



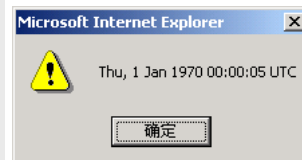
### 创建日期对象

```
//方法1: 不指定参数  
var nowd1=new Date();  
alert(nowd1.toLocaleString());  
//方法2: 参数为日期字符串  
var nowd2=new Date("2004/3/20 11:12");  
alert(nowd2.toLocaleString());  
var nowd3=new Date("04/03/20 11:12");  
alert(nowd3.toLocaleString());
```



### 创建日期对象

```
//方法3: 参数为毫秒数  
var nowd3=new Date(5000);  
alert(nowd3.toLocaleString());  
alert(nowd3.toUTCString());  
  
//方法4: 参数为年月日小时分钟秒毫秒  
var nowd4=new Date(2004,2,20,11,12,0,300);  
alert(nowd4.toLocaleString());  
//毫秒并不直接显示
```

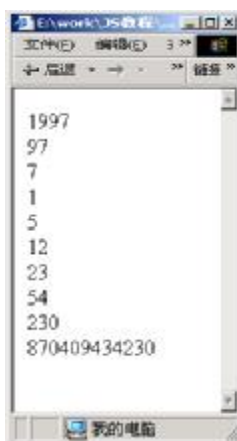


月份参数从0~11, 所以这里2对应3月份

## u 获取日期和时间

- | getDate() 获取日
- | getDay () 获取星期
- | getMonth () 获取月
- | getFullYear () 获取完整年份
- | getYear () 获取年
- | getHours () 获取小时
- | getMinutes () 获取分钟
- | getSeconds () 获取秒
- | getMilliseconds () 获取毫秒
- | getTime () 返回累计毫秒数(从1970/1/1午夜)

## u 程序示范



```
function writeln(mystr){
    document.write(mystr+"<br>");
}
var x=new Date(1997,7,1,12,23,54,230)
//1997年8月1日12点23分54秒230毫秒
writeln(x.getFullYear()); //返回年1997
writeln(x.getYear()); //返回年97
writeln(x.getMonth()); //返回月7
writeln(x.getDate()); //返回日1
writeln(x.getDay()); //返回星期5
writeln(x.getHours()); //返回小时12
writeln(x.getMinutes()); //返回分钟23
writeln(x.getSeconds()); //返回秒54
writeln(x.getMilliseconds()); //返回毫秒230
writeln(x.getTime()); //返回累计870409434230
```

## └ 设置日期和时间

- | setDate(day\_of\_month)                    设置日
- | setMonth (month)                    设置月
- | setFullYear (year)                    设置年
- | setHours (hour)                    设置小时
- | setMinutes (minute)                    设置分钟
- | setSeconds (second)                    设置秒
- | setMilliseconds (ms)                    设置毫秒(0-999)
- | setTime (allms) 设置累计毫秒(从1970/1/1午夜)

## └ 程序示范



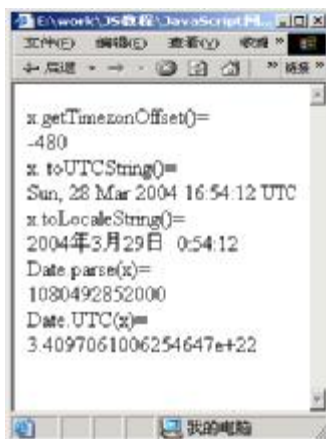
```
var x=new Date();
x.setFullYear (1997); //设置年1997
x.setMonth(7);        //设置月7
x.setDate(1);        //设置日1
x.setHours(5);        //设置小时5
x.setMinutes(12);     //设置分钟12
x.setSeconds(54);     //设置秒54
x.setMilliseconds(230);     //设置毫秒230
document.write(x.toLocaleString()+"<br>");
//返回1997年8月1日5点12分54秒

x.setTime(870409430000); //设置累计毫秒数
document.write(x.toLocaleString()+"<br>");
//返回1997年8月1日12点23分50秒
```

## u 日期和时间的转换

- | **getTimezoneOffset()**
  - u 返回本地时间与GMT的时间差，以分钟为单位
- | **toUTCString()**
  - u 返回国际标准时间字符串
- | **toLocaleString()**
  - u 返回本地格式时间字符串
- | **Date.parse(x)**
  - u 返回累计毫秒数(从1970/1/1午夜到本地时间)
- | **Date.UTC(x)**
  - u 返回累计毫秒数(从1970/1/1午夜到国际时间)

## u 程序示范



```
function writeln(mystr){
    document.write(mystr+"<br>");
}
var x=new Date();
writeln("x.getTimezoneOffset()=");
writeln(x.getTimezoneOffset());//时间差
writeln("x.toUTCString()=");
writeln(x.toUTCString());//国际时间
writeln("x.toLocaleString()=");
writeln(x.toLocaleString());//本地时间
writeln("Date.parse(x)=");
writeln(Date.parse(x));//本地时间毫秒数
writeln("Date.UTC(x)=");
writeln(Date.UTC(x));//UTC毫秒数
```

- └ JavaScript概述
- └ JavaScript基础语法
- └ JavaScript常用内置对象
- └ JavaScript常用DHTML对象
- └ JavaScript高级技巧

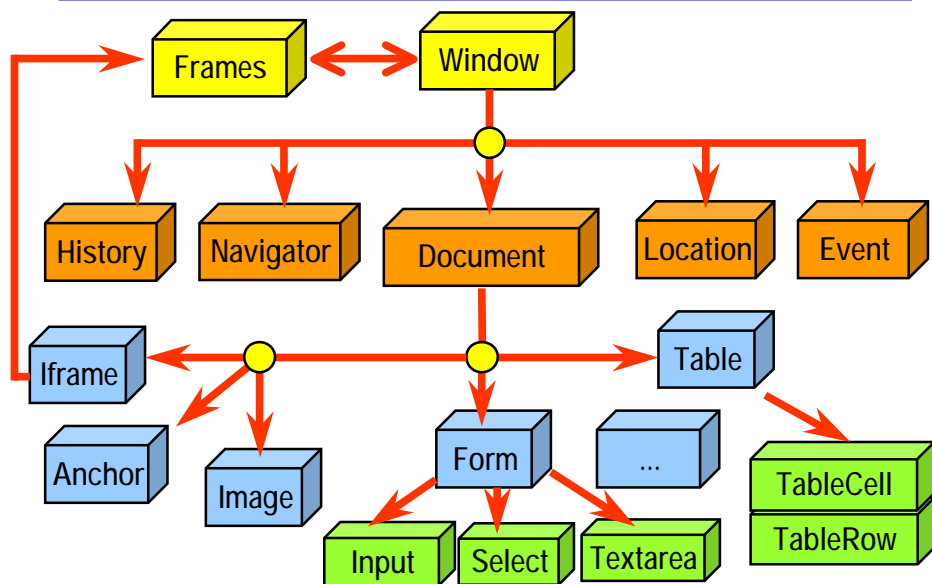
- └ DHTML的定义
  - ┆ 使用JavaScript和CSS级联样式表操作HTML创造出各种动态视觉效果统称为DHTML
  - ┆ DHTML是一种浏览器端的动态网页技术

### └ DHTML的功能

- ┆ 动态改变字体大小和字体颜色
- ┆ 动态设定文档元素的位置、内容，甚至隐藏和显示元素
- ┆ 可以通过事件响应机制制作动态折叠的树形结构和菜单
- ┆ 可以通过定时器制作时钟、日历
- ┆ 可以弹出对话框与用户进行交互
- ┆ 可以通过表单提交用户填写的信息
- ┆ 通过动态样式表可以设定更多的显示效果
- ┆ ...

### └ DHTML对象模型

- ┆ 将HTML标记、属性和CSS样式都对象化
- ┆ 可以动态存取HTML文档中的所有元素
- ┆ 可以使用属性name或id来存取或标记对象
- ┆ 改变元素内容或样式后浏览器中显示效果即时更新
- ┆ DHTML对象模型包括浏览器对象模型和Document对象模型



北京: 010-62196102 上海: 61202630

143

外企的师资、外企的技术、外企的品质

### └ DHTML DOM与W3C DOM的比较

比较项目	DHTML DOM	W3C DOM
概念	DHTML中的 Document对象模型	标准的树形结构 文档操作接口
浏览器支持	IE4.0以上	IE5.0以上
实现方法	对象数组 Document.all	树形节点对象 Node.Element
操作语言	JavaScript	JavaScript、 Java、C++等
文档对象	HTML	HTML、XML

北京: 010-62196102 上海: 61202630

144

外企的师资、外企的技术、外企的品质



## 常用属性

名称	功能说明
document	对象，代表窗口中显示的HTML文档
frames	窗口中框架对象的数组
history	对象，代表浏览过窗口的历史记录
location	对象，代表窗口文件地址，修改属性可以调入新的网页
defaultStatus, status	窗口的状态栏信息
closed	窗口是否关闭，关闭时该值为true
name	窗口名称，用于标识该窗口对象

## 常用属性

名称	功能说明
opener	对象，是指打开当前窗口的window对象，如果当前窗口被用户打开，则它的值为null
parent	对象，当前窗口是框架页时指的是包含该框架页的上一级框架窗口
top	对象，当前窗口是框架页时指的是包含该框架页的最外部的框架窗口
self	对象，指当前Window对象
window	对象，指当前Window对象，同self

## 常用方法

名称	功能说明
alert(),confirm,prompt()	弹出简单对话框
close(),open()	关闭、打开窗口
print()	打印窗口中网页的内容
focus(),blur()	请求或放弃窗口为当前操作窗口
moveBy(),moveTo()	移动窗口
resizeBy(),resizeTo()	调整窗口大小
scrollBy(),scrollTo()	滚动窗口中网页的内容
setInterval(),clearInterval()	设置或取消周期执行的定时器
setTimeout(),clearTimeout()	设置或取消一次性执行的定时器

## 主要功能

- ▮ 窗口的打开和关闭
- ▮ 对话框
- ▮ 状态栏
- ▮ 定时器
- ▮ 内容滚动
- ▮ 调整窗口大小和位置
- ▮ Screen对象
- ▮ History对象
- ▮ Navigator对象
- ▮ Location对象

书写格式	功能说明
<code>window.open(url,name,config)</code>	打开新窗口 url为打开的超链接 name为窗口的名称 config为窗口的配置参数 返回新窗口对象
<code>window.close()</code>	关闭窗口

#### config 参数具体元素

- | menubar 菜单条
- | toolbar 工具条
- | location 地址栏
- | directories 链接
- | status 状态栏
- | scrollbars 滚动条
- | resizable 可调整大小
- | width 窗口宽, 以像素为单位
- | height 窗口高, 以像素为单位

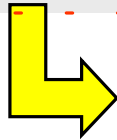
参数值为  
yes或no

参数值为  
数字 值

## 程序示范：打开google搜索窗口

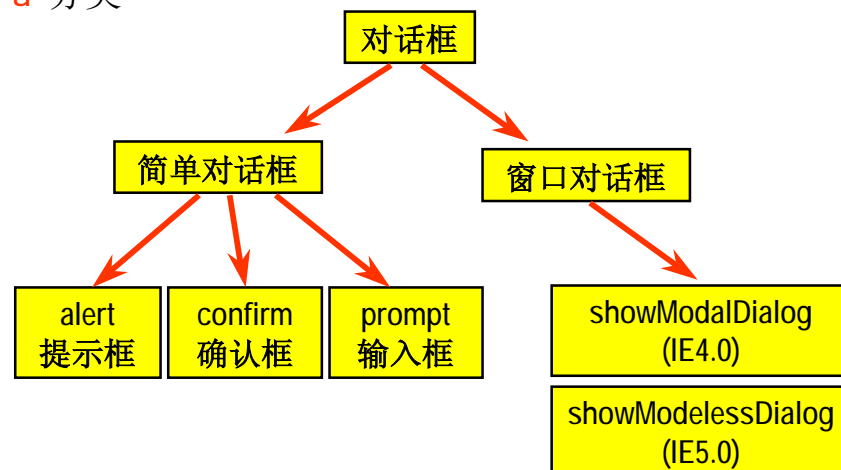
```
var config= 'menubar=yes,toolbar=no,location=no, ';  
config += 'directories=no,status=yes, ';  
config += 'scrollbars=yes,resizable=yes, ';  
config += 'width=500,height=300';
```

```
var openurl=" http://www.google.com";  
window.open(openurl,"popwin",config);  
//仅仅打开窗口
```



```
var mywin=window.open(openurl,"popwin",config);  
mywin.close(); //关闭打开的窗口
```

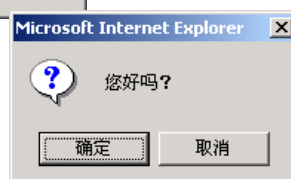
## 分类



书写格式	功能说明
alert(str)	提示对话框，显示str字符串的内容 按[确定]关闭对话框
confirm(str)	确认对话框，显示str字符串的内容 按[确定]按钮返回true，[取消]返回false
prompt(str,value)	输入对话框，采用文本框输入信息 按[确定]按钮返回输入值，[取消]关闭

显示效果比较:

```
alert("您好!");  
confirm("您好吗?");  
prompt("您贵姓?","陈");
```



返回值比较:

返回值为输入字符串

```
var firstname=prompt("您贵姓? ","陈");  
if (confirm("您确定?")==true) {  
    alert(firstname+"先生, 您好! ");  
}
```

返回值为true或false


不返回值

书写格式	功能说明
showModalDialog(url,arguments,config)	IE4或更高版本支持该方法
showModelessDialog(url,arguments,config)	IE5或更高版本支持该方法

参数说明

- | url 打开链接
- | arguments 传入参数
- | config 窗口配置参数

## u config 外观配置参数

- | status 状态栏
- | resizable 可调整大小
- | help 是否显示标题栏中的  按钮
- | center 是否显示在桌面正中间
- | dialogWidth 对话框宽
- | dialogHeight 对话框高
- | dialogTop 对话框左上角的y坐标
- | dialogLeft 对话框左上角的x坐标

参数值为  
yes或no值为数字  
单位为像素

## u 程序示范: 调用窗口对话框的方法

```
...  
<script language="javascript">  
function showDialog(){  
    var config ='dialogWidth:320px;dialogHeight:180px;';  
    config += 'dialogTop:140px;dialogLeft:250px;';  
    config += 'center:no;help:no;resizable:no;status:no';  
    showModalDialog("test4-1a.htm",input1,config);  
}  
</script>  
<input type="text" id="input1" value="" readonly>  
<input type="button" id="selectBtn"  
    onclick="showDialog()" value="选择日期">  
...
```

传入input1对象

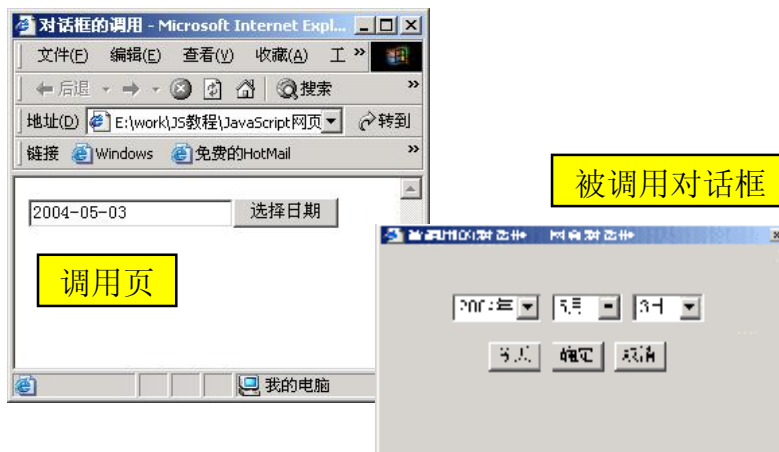


## 程序示范：窗口对话框页面的编写

```
<script language="javascript">
//确定按钮事件，更新输入框中的值
function ok(){
    var theInput=dialogArguments;
    ...
    theInput.value=theYear+"-"+theMonth+"-"+theDate;
    window.close();
}
</script>
<input type="button" name="okbtn" value="确定" onclick="ok()">
```

设置传入对象  
input1的值

## 运行结果：选择日期对话框

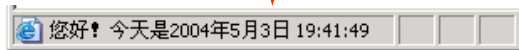




书写格式	功能说明
window.status	状态栏中的字符串信息 允许进行设置或读取

```
window.status="hello";
```

```
var str="您好！今天是"+(new Date()).toLocaleString();  
window.status=str;
```



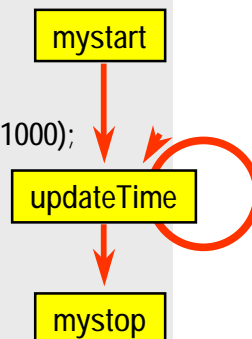
书写格式	功能说明
tID=setInterval(exp,time)	周期性触发执行代码exp exp为字符串格式的语句 time为时间周期，单位为毫秒 返回已经启动的定时器
clearInterval(tID)	停止启动的定时器
tID=setTimeout(exp,time)	一次性触发执行代码exp exp为字符串格式的语句 time为间隔时间，单位为毫秒 返回已经启动的定时器
clearTimeout(tID)	停止启动的定时器

u 定时器的实际运用

- | 网页动态时钟
- | 制作倒计时
- | 跑马灯效果
- | 幻灯片效果
- | 自动滚屏阅读
- | 制作网页小游戏
- | .....

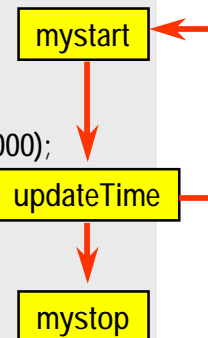
u 程序示范：网页动态时钟（采用setInterval）

```
...  
var timerID=null;  
function updateTime(){ //更新状态栏显示当前时间  
    var now=(new Date()).toLocaleString();  
    window.status="当前时间："+now;  
}  
function mystart(){ //启动定时器  
    timerID=window.setInterval("updateTime()",1000);  
}  
function mystop(){ //停止定时器  
    window.clearInterval(timerID);  
}  
...
```



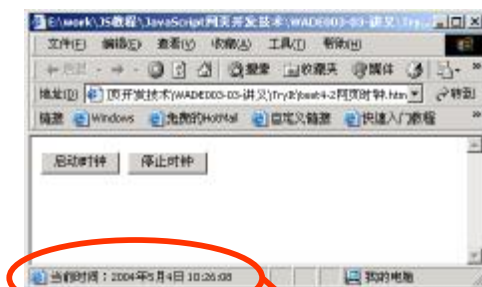
## u 程序示范：网页动态时钟（采用setTimeout）

```
...  
var timerID=null;  
function updateTime(){ //更新状态栏显示当前时间  
    var now=(new Date()).toLocaleString();  
    window.status="当前时间： "+now;  
    mystart();  
}  
function mystart(){ //启动定时器  
    timerID=window.setTimeout("updateTime()",1000);  
}  
function mystop(){ //停止定时器  
    window.clearTimeout(timerID);  
}
```



北京: 010-62196102 上海: 61202630 165 外企的师资、外企的技术、外企的品质

## u 运行结果：网页动态时钟



状态栏中的时间信息将每秒刷新一次

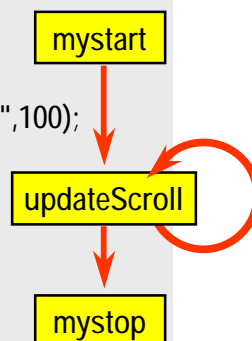
当前时间：2004年5月4日 10:25:29

北京: 010-62196102 上海: 61202630 166 外企的师资、外企的技术、外企的品质

书写格式	功能说明
window.scroll(x,y)	滚动窗口到指定位置 单位为像素
window.scrollTo(x,y)	同scroll方法
window.scrollBy(ax,ay)	从当前位置开始，向右滚动ax像素， 向下滚动ay像素

## u 程序示范：自动滚屏阅读

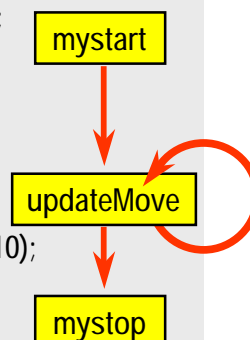
```
...  
var timerID=null;  
function updateScroll(){ //更新滚动位置  
    window.scrollBy(0,1);  
}  
function mystart(){ //启动定时器  
    timerID=window.setInterval("updateScroll()",100);  
}  
function mystop(){ //停止定时器  
    window.clearInterval(timerID);  
}  
...
```



书写格式	功能说明
window.moveTo(x,y)	移动窗口到指定位置 单位为像素
window.moveBy(ax,ay)	向右移动ax像素 向下移动ay像素 参数为负数表示反方向移动
window.resizeTo(width,height)	调整窗口大小为指定大小
window.resizeBy(ax,ay)	放大或缩小窗口 参数为负数表示缩小

## 程序示范: 会摇头的窗口

```
var STEPSIZE=10;//定义移动的步长
var timerID=null,tempx=STEPSIZE;
function updateMove(){ //更新移动位置
  if (window.screenLeft>500) tempx=-STEPSIZE;
  if (window.screenLeft<100) tempx=STEPSIZE;
  window.moveBy(tempx,0);
}
function mystart(){ //启动定时器
  window.resizeTo(400,300);
  timerID=window.setInterval("updateMove()",10);
}
function mystop(){ //停止定时器
  window.clearInterval(timerID);
}
```



## u 对象介绍

- 属于window的子对象
- 常用于获取屏幕的分辨率和色彩

书写格式	功能说明
screen.width	屏幕分辨率的宽度，例如1024*768分辨率下宽度为1024
screen.height	返回上一页
screen.availWidth	屏幕中可用的宽
screen.availHeight	屏幕中可用的高
screen.colorDepth	屏幕的色彩数

## u 应用例子：窗口最大化

```
window.moveTo(0,0);  
window.resizeTo(screen.availWidth,screen.availHeight);
```



采用screen对象的分辨率属性和resizeTo方法来动态确定窗口最大长度和宽度

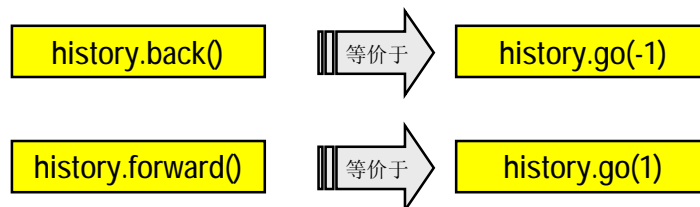
## u 对象介绍

- 属于window的子对象
- 常用于返回到已经访问过的页面

书写格式	功能说明
history.length	历史记录数
history.back()	返回上一页
history.forward	前进一页
history.go(num)	前进或后退num页，num为0时表示页面刷新

## u 应用例子: 网页导航按钮

```
<input type="button" value="返回" onclick="history.back()">  
<input type="button" value="前进" onclick="history.forward()">  
  
<input type="button" value="刷新" onclick="history.go(0)">
```



### u Navigator对象

- | 属于window的子对象，只有属性
- | 常用于获取客户端浏览器和操作系统信息
- | 常用的属性
  - u **appName**      浏览器的名称
  - u **appVersion**      浏览器的版本和操作系统名称
  - u **userLanguage**      用户语系
  - u **cookieEnable**      是否支持cookie

### u 获取Navigator对象的所有属性

```
var showtext = "Navigator对象属性列表: \n";
for (var propname in navigator) {
    showtext += propname + ": " + navigator[propname] + "\n";
}
alert(showtext);
```

```
Navigator对象属性列表:
appName: Mozilla
appName: Microsoft Internet Explorer
appMinorVersion: ;SP1;Q832894;Q330994;Q837009;Q831167;
cpuClass: x86
platform: Win32
plugins:
opsProfile:
userProfile:
systemLanguage: zh-cn
userLanguage: zh-cn
appVersion: 4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
userAgent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
onLine: true
cookieEnabled: true
mimeTypes:
```



### u Mozilla下运行的结果



Navigator对象属性列表：

```
appCodeName: Mozilla
appName: Netscape
appVersion: 5.0 (Windows; zh-CN)
language: zh-CN
mimeTypes: [object MimeTypeArray]
platform: Win32
oscpu: Windows NT 5.0
vendor:
vendorSub:
product: Gecko
productSub: 20040113
plugins: [object PluginArray]
securityPolicy:
userAgent: Mozilla/5.0 (Windows; U; Windows NT 5.0; zh-CN; rv:1.6) Gecko/20040113
cookieEnabled: true
```

北京: 010-62196102 上海: 61202630 177 外企的师资、外企的技术、外企的品质

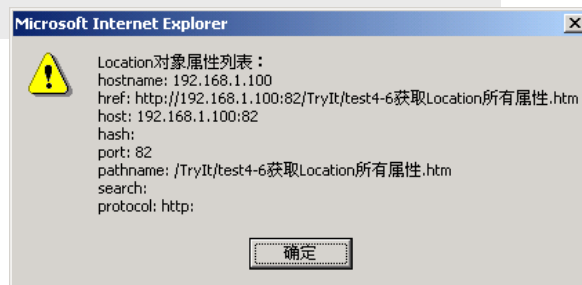
### u Location对象

- | 属于window的子对象
- | 常用于获取和改变当前浏览的网址
- | 常用的属性
  - u href 当前窗口正在浏览的网页地址
  - u replace(url) 转向到url网页地址
  - u reload() 重新载入当前网址，同按下刷新按钮

北京: 010-62196102 上海: 61202630 178 外企的师资、外企的技术、外企的品质

## u 获取Location对象的所有属性

```
var showtext = "Location对象属性列表：\n";  
for (var proptime in location) {  
    showtext += proptime + ": " + location[proptime] + "\n";  
}  
alert(showtext);
```



## u 常用字符串属性

名称	功能说明
title	文档标题，通过窗口标题栏显示
URL, location	文档的载入链接
referrer	链接到该文档的前一文档链接，只读
lastModified	文档最后修改日期的字符串，只读
bgColor, fgColor	文档背景和缺省字体颜色
alinkColor	链接按下激活后的颜色
linkColor	未访问过链接的颜色
vlinkColor	已经访问过链接的颜色

## 程序示范

- 获取上次修改时间
- 标题栏动态修改
- 背景颜色调整



```
document.write("最后修改时间: "+document.lastModified);  
document.title="欢迎进入JavaScript世界";  
document.bgColor="#EFEFEF"; //设置背景为灰色  
document.fgColor="red"; //设置字体为红色
```

## 常用对象属性

名称	功能说明
forms[ ]	Form对象数组, 存放文档中所有表单
images[ ]	Image对象数组, 存放文档所有图片
anchors[ ]	链接定义数组, 存放文档中所有链接
applets[ ]	存放文档中Java小程序的数组
cookie	子对象, 用于在客户端存储信息
all[ ]	存放文档中所有对象的数组, IE支持

## └ 数组对象的调用

- ┆ 根据对象索引号，如forms[0]代表文档中第一个表单
- ┆ 根据对象名称，如forms["form1"]代表名称为form1的表单
- ┆ 直接对象调用，如document.form1

```
<form name="form1">
<input type="text" name="input1">
<input type="submit" value="提交">
</form>
<script language="JavaScript">
document.form1.input1.value=(new Date()).toLocaleString();
</script>
```

一般情况下采用直接对象调用方式调用表单

## └ 关键方法

名称	功能说明
close()	关闭open()方法打开的文档
open()	初始化文档对象
write(str)	将文本附加到当前打开的文档
writeln(str)	将文本附加到当前打开的文档并加上一个换行符

程序示范：输出内容到弹出新窗口

```
var myWin=window.open(); //打开新窗口  
myWin.document.open(); //初始化显示文档
```

```
myWin.document.write("直接输出；");  
myWin.document.writeln("1.输出后换行");  
myWin.document.writeln("2.输出后换行");  
myWin.document.writeln("3.输出后换行");
```

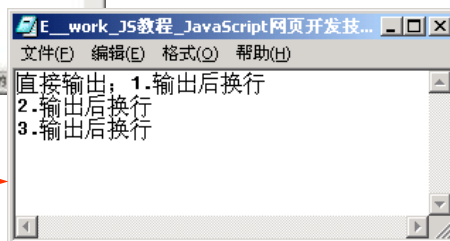
```
myWin.document.close(); //关闭文档
```

如果不指定Window对象  
则代表直接操作当前窗  
口的Document对象

程序结果：



writeln输出的换行符在  
原代码中可以看到



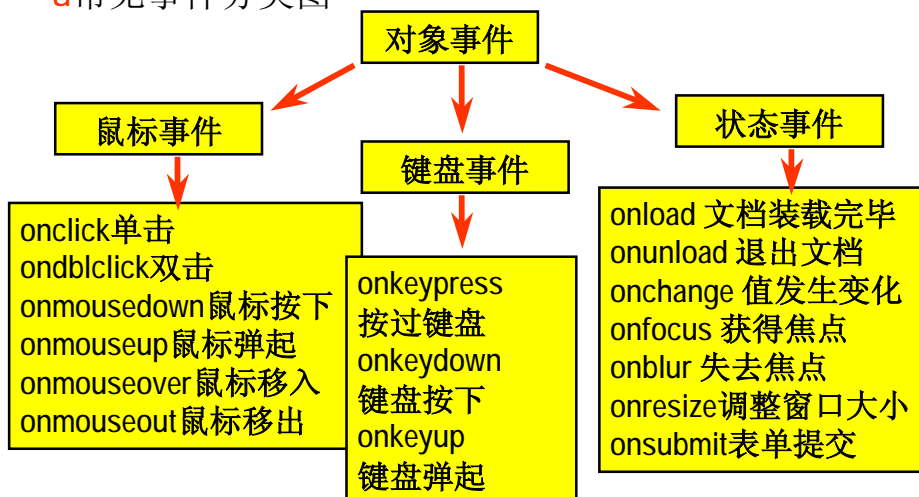
### 对象事件概念

- 指DHTML对象在状态改变、操作鼠标或键盘时触发的动作

### 对象事件的分类

- 鼠标事件：针对鼠标单击、双击、移动等动作
- 键盘事件：针对按下键盘产生的动作
- 状态改变事件：在载入文档、退出、获取焦点、失去焦点、提交、变化等动作时产生的事件

### 常见事件分类图



### u Event 事件对象

- | 事件触发后将会产生一个Event对象
- | Event对象记录事件发生时的鼠标位置、键盘按钮状态和触发对象等信息

### u 主要的属性

#### | clientX、clientY

事件触发时鼠标光标相对浏览器窗口的坐标

#### | screenX、screenY

事件触发时相对客户端屏幕的位置坐标

#### | offsetX、offsetY

事件触发时相对引发事件标记对象的位置

#### | x、y

事件触发时鼠标光标相对父组件的位置坐标

### u 主要的属性

- | **srcElement** 触发该事件的标记对象
- | **button** 鼠标按下的键（1左键、2右键、4中键）
- | **keyCode** 键盘按键的Unicode码
- | **altKey/ctrlKey/shiftKey** 键盘的alt、ctrl、shift键是否按下

（以下专门针对onmouseover和onmouseout事件）

- | **fromElement** 鼠标原来所处标记对象
- | **toElement** 鼠标现在所处标记对象

### u 属性定义中直接处理事件

定义进入、退出文档和单击按钮事件

<!--在属性定义中直接处理对象事件-->

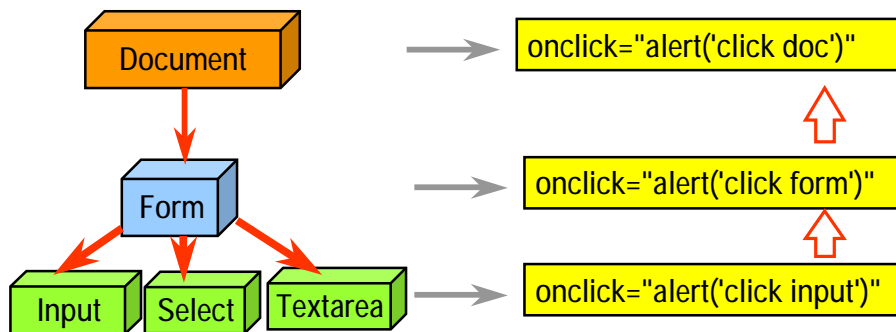
```
<body onload="alert('hello body')" onunload="alert('goodbye')">  
<input type="button" value="单击" onclick="alert('hello button')">  
</body>
```



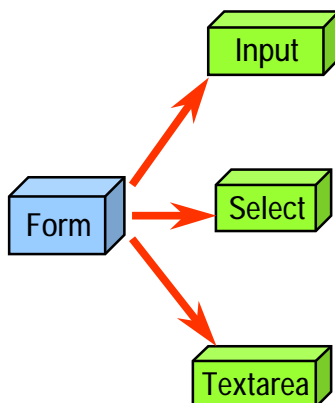


## IE的冒泡事件处理机制

当处于DHTML对象模型底部对象事件发生时，会依次激活上面对象定义的同类事件处理



## 表单元素



```
<input type="text" name="a1" value="1">
```

type包括 text、button、submit、reset、checkbox、radio、hidden等

```
<select name="s1">
<option value="1">选择一</option>
<option value="2">选择二</option>
</select>
```

```
<textarea name="t1">
初始内容
</textarea>
```

u 表单验证

```
<script language="JavaScript">
function check(){ //检查t1输入是否为空
    if (form1.t1.value!=""){
        form1.submit();
    }else{
        form1.t1.focus();
    } //如果t1值为空则自动将焦点设定到t1输入框
}
</script>
<form name="form1">
    姓名: <input type="text" name="t1" value="">
    <input type="button" name="b1" onclick="check()" value="提交">
</form>
```



u 介绍

- Cookie是Web浏览器用来存储少量数据的存储方式
- Cookie是Document对象的一个字符串类型属性

u 主要功能

- 用来存储用户输入的信息，在下次输入时自动调出来，例如存储用户名和口令
- 用来记录服务器端的少量字符串变量
- 用来记录用户访问同一网页的次数
- 商务网站购物车功能的实现

### Cookie的JavaScript存取方法

- 通过document.cookie直接存取
- cookie的属性可以设定终止日期(expires)、路径(path)、域(domain)、是否安全(secure)
- cookie属性之间需要用分号分开

```
var expdate=(new Date("2005-05-01")).toUTCString;  
document.cookie="username=a1;expires="+expdate;  
document.cookie="username=a1;expires="+expdate;  
alert(document.cookie);
```

expires必须是UTC国际  
标准时间格式字符串



### 程序示范：通用cookie存取函数

```
// 设定cookie的函数  
function SetCookie(name, value,expires,path,domain,secure) {  
    //设定编码后的cookie值  
    var t1= name + "=" + escape (value);  
    if (expires) t1 += "; expires=" + (new Date(expires)).toGMTString();  
    if (path) t1 += "; path=" + path;  
    if (domain) t1 += "; domain=" + domain;  
    if (secure) t1 += "; secure=" + secure;  
    document.cookie=t1;  
}
```

### 程序示范：通用cookie存取函数

```
//通过名称获取cookie
function GetCookie(name) {
    var arg = name + "=";
    var tempstr=document.cookie;
    var pos1 = tempstr.indexOf(arg);
    var pos2 = tempstr.indexOf(";",pos1);
    if (pos1== -1){return null;}
    if (pos2== -1){pos2=tempstr.length;}
    tempstr=tempstr.substring((pos1+arg.length),pos2);
    return unescape(tempstr);
}
```

### CSS样式表

- 链入外部样式表文件 (Linking to a Style Sheet)
  - ü 先建立外部样式表文件 (.css)，然后使用HTML的link对象
- 定义内部样式块对象 (Embedding a Style Block)
  - ü 可以在HTML文档的<HTML>和<BODY>标记之间插入一个<STYLE>...</STYLE>块对象
- 内联定义 (Inline Styles)
  - ü 内联定义是指在对象的标记内使用对象的style属性直接定义少数的样式表属性

### 三种样式表的不同定义方法

```
<link rel=stylesheet href="test.css" type="text/css">
```

```
<style type="text/css">
<!--
body {font: 10pt}
p {font: 10pt; color: black}
-->
</style>
```

外部CSS文件无需加  
<style>标签

```
<!--
body {font: 10pt}
p {font: 10pt; color: black}
-->
```

```
<p style="margin-left: 0.5in; margin-right:0.5in">测试</p>
```

直接作为属性设定

北京: 010-62196102 上海: 61202630

201

外企的师资、外企的技术、外企的品质

### 动态样式表

- 指的是在JavaScript中直接对对象的style属性进行设定
- HTML组件中的style属性将立即起作用

```
<div style="font-size:36pt;font-family: 黑体"
onmouseover="this.style.color='#FF0000';"
onmouseout="this.style.color='#000000';" >
鼠标移过来，马上变颜色
</div>
```



北京: 010-62196102 上海: 61202630

202

外企的师资、外企的技术、外企的品质

- └ JavaScript概述
- └ JavaScript基础语法
- └ JavaScript常用内置对象
- └ JavaScript常用DHTML对象
- └ JavaScript高级技巧

- └ 正则表达式的概念
  - ┆ 是一种模式匹配下文本替换、搜索、提取的强有力工具
  - ┆ 在JavaScript下是一个用于处理字符串匹配的对象
- └ 正则表达式的功能
  - ┆ 查找文本中的字符串
  - ┆ 替换文本中匹配的字符串
  - ┆ 对表单输入值进行校验

### Regular Expression对象定义

- rgExp=/pattern/flags
- rgExp=new RegExp("pattern",["flags"])

### flags标识有以下几个:

- g: 设定当前匹配为全局模式
- i: 忽略匹配中的大小写检测
- m: 多行搜索模式

### flags使用范例:

```
var re = /ow/ig;  
var str = "HelloWorld,How are you!";  
var arr = str.match(re);  
alert(arr);
```

flag可以组合或单独使用

使用不同的flag, 结果也不同

```
var re = /ow/ig;
```



```
var re = /ow/g;
```



```
var re = /ow/i;
```



## Regular Expression对象的属性

| 属性        | 说明           | IE支持 |
|-----------|--------------|------|
| source    | 正则表达式匹配模式    | IE4  |
| lastIndex | 匹配字符串末尾字符的位置 | IE4  |

## Regular Expression对象的方法

| 方法                               | 说明                         |
|----------------------------------|----------------------------|
| RegExp.compile(pattern, [flags]) | 将RegExp转化为内部格式, 以加快匹配的执行   |
| RegExp.exec(str)                 | 按照RegExp的匹配模式对str字符串进行匹配查找 |
| RegExp.test(str)                 | 匹配字符串末尾字符的位置               |



程序示范：验证电子邮件地址是否合法

```
function validateEmail(emailStr){  
    var re=/^[w.-]+@[0-9a-z][w.-]+\.[a-z]{2,3}$/i;  
    if(re.test(emailStr)) {  
        alert("有效email地址!");  
        return true;  
    } else {  
        alert("无效email地址!");  
        return false;  
    }  
}  
validateEmail("yourname@mailhost.com");
```



单个字符匹配模式

| 字符    | 匹配    |
|-------|-------|
| 字母和数字 | 自身    |
| \n    | 换行符   |
| \r    | 回车符   |
| \o    | NUL字符 |
| \t    | 制表符   |
| \v    | 垂直制表符 |

### 多个字符匹配模式

| 字符     | 匹配               | 例子                   |
|--------|------------------|----------------------|
| [...]  | 位于括号内的任意字符       | /[xyz]/匹配x,y,z任一字符   |
| [^...] | 不在括号中的任意字符       | /[^xyz]/匹配任何非x,y,z字符 |
| [:]    | 除了换行符外任意字符       |                      |
| \w     | 任何ASCII单字符       |                      |
| \W     | 任何ASCII非单字符      |                      |
| \s     | 任何Unicode空白符     |                      |
| \S     | 任何非Unicode空白符    |                      |
| \d     | 任何ASCII码单数字      | /\d\d/可以匹配一个两位数      |
| \D     | 除了ASCII数字之外的任何字符 |                      |

### 重复匹配模式

| 字符    | 匹配                | 例子                   |
|-------|-------------------|----------------------|
| {n,m} | 匹配前一项至少n次,但不能超过m次 | \d{2,3}匹配10至999数字字符串 |
| {n,}  | 匹配前一项至少n次或更多次     | \d{2,}匹配10以上的数字字符串   |
| {n}   | 匹配前一项恰好n次         | \w{3}匹配三个ASCII字符     |
| ?     | 匹配前一项0或1次         | \w?匹配一个ASCII字符       |
| +     | 匹配前一项1或多次         | \w+匹配1个或多个字符         |
| *     | 匹配前一项0或多次         | \w*匹配0个或多个字符         |

## └ RegExp对象

- ┆ 是一个自动创建的全局对象
- ┆ 每次成功匹配操作结果信息保存在该对象属性中

## └ RegExp对象与Regular Expression对象的区别

- ┆ Regular Expression对象在匹配前需要指定匹配模式，即创建一个Regular Expression对象的实例
  - ┆ 例如：rgExp=new RegExp();
- ┆ RegExp对象无需创建，它是一个固有的全局对象

## └ RegExp对象的属性：最近一次成功匹配的结果信息

| 属性        | 说明           | IE支持 |
|-----------|--------------|------|
| input     | 被搜索的目标字符串    | IE4  |
| index     | 匹配字符串首字符的位置  | IE4  |
| lastIndex | 匹配字符串末尾字符的位置 | IE4  |
| \$1 - \$9 | 匹配中最开始的9个子匹配 | IE4  |

RegExp对象的属性使用范例：

```
var s;  
var re = /(w{3})\.(\w+)\.(\w{2,3})\.(\w{2,3})/ig;  
var str = "http://www.itins.com.cn";  
var arr = re.exec(str);  
s = "arr 保存: " + arr + "<BR>";  
s += "$1 保存: " + RegExp.$1 + "<BR>";  
s += "$2 保存: " + RegExp.$2 + "<BR>";  
s += "$3 保存: " + RegExp.$3 + "<BR>";  
s += "$4 保存: " + RegExp.$4 + "<BR>";  
document.write(s);
```



字符串对象中正则表达式应用模式匹配的方法

| 方法                                    | 说明   |
|---------------------------------------|--|
| stringObj.match(rgExp)                | 根据正则表达式模式查找字符串中的匹配字符项，返回值同exec方法                     |
| stringObj.replace(rgExp, replaceText) | 返回一个字符串，即将stringObj中符合rgExp模式匹配的字符串替换成replaceText后返回 |
| stringObj.search(rgExp)               | 返回第一个匹配到的子字符串的位置                                     |

### 字符串高级查找替换操作使用范例：

```
var s;  
var re = /(w{3})\.(\w+)\.(\w{2,3})\.(\w{2,3})/ig;  
var str = "http://www.itins.com.cn";  
var arr = str.match(re);  
s = "arr 保存: " + arr + "<BR>";  
s += "$1 保存: " + RegExp.$1 + "<BR>";  
s += "$2 保存: " + RegExp.$2 + "<BR>";  
s += "$3 保存: " + RegExp.$3 + "<BR>";  
s += "$4 保存: " + RegExp.$4 + "<BR>";  
document.write(s);
```



### 错误处理的相关概念

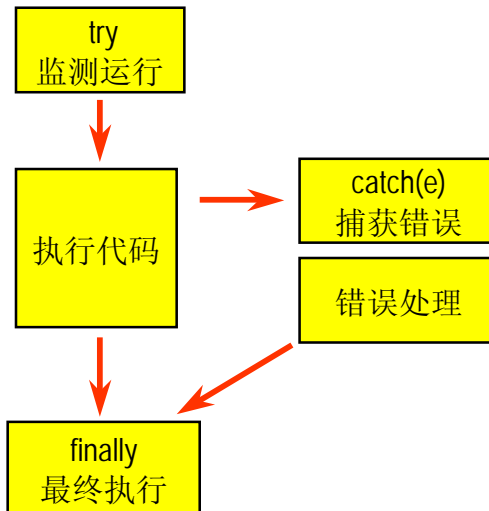
- JavaScript解释器在遇到程序上的错误时会抛出错误信息
- 错误处理是自动获取运行中的错误信息并进行处理的过程
- Error是JavaScript提供的一个错误信息存放对象

### 错误处理的必要性

- 通过对错误的截获可以让程序自动处理发生意外的情况
- 可以让程序对无关紧要的错误进行忽略，保证程序运行
- 错误处理可以让程序更加健壮
- 在优秀的程序代码中，错误处理是不可少的

## u 错误处理的程序结构

```
try{  
    //执行代码  
    ...  
}  
catch(e){  
    //错误处理  
    ...  
}  
finally{  
    //最终执行代码  
}
```



## u Error对象的属性

| 属性          | 说明                     |
|-------------|------------------------|
| number      | 错误码，是一个32位值，当前只用到了后16位 |
| message     | 用于存放错误描述信息             |
| description | 用于存放错误描述信息             |

## u 输出程序运行出错信息

```
var x=0;
try{
    document.write(X);
    //大小写错误
}
catch(e){
    document.write("错误代码:" + (e.number&0xFFFF) + "<br>");
    document.write("错误message:"+e.message+"<br>");
    document.write("错误description:"+e.description+"<br>");
}
```



北京: 010-62196102 上海: 61202630 221 外企的师资、外企的技术、外企的品质

## u 文件系统对象（FileSystemObject）是一个ActiveX对象

- l 用于获取浏览器所在客户端机器的磁盘驱动信息
- l 用于添加、修改或删除文件和文件夹
- l 用于读写文本文件的内容

## u 创建文件系统对象的方法

- l `var fso=new ActiveXObject("Scripting.FileSystemObject")`

北京: 010-62196102 上海: 61202630 222 外企的师资、外企的技术、外企的品质

### 安全性限制介绍

- 在本地网页中的JavaScript创建文件系统对象的时候系统会警告
- 如果网页来自Internet, 缺省浏览器会禁止执行文件操作
- 可以通过设置IE来允许进行文件操作
- 访问站点如果被设置为安全站点那么文件操作将被允许



### FileSystemObject对象属性

| 属性         | 说明     |
|------------|--------|
| Drive      | 磁盘对象   |
| Drives     | 本地磁盘集合 |
| Folder     | 文件夹对象  |
| File       | 文件对象   |
| TextStream | 文本对象   |



u 获取FileSystemObject子对象:

l 创建新的文件或文件夹:

- u CreateFolder(folderPathName) 指定路径创建文件夹对象
- u CreateTextFile(filePathName,flag) 在指定位置创建文件,flag为是否覆盖已经存在文件

l 如果创建成功, 可以返回对应的文件或文件夹对象

//例如以下是创建新文件夹的例子:

```
var fso, fldr;  
fso = ActiveXObject("Scripting.FileSystemObject");  
fldr = fso.CreateFolder("C:\\Test");  
alert("已经创建新文件夹: " + fldr.Name);
```

u 获取FileSystemObject子对象:

l FileSystemObject 对象中的 "get"系列方法可以访问现有驱动器、文件或文件夹:

- u GetDrive(driverName) 通过驱动器名获得驱动器对象
- u GetFolder(folderPathName) 通过路径获得文件夹对象

//例如以下是获取文件对象File的例子:

```
var fso, f1;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
f1 = fso.GetFile("c:\\test.txt");
```

对象

## u Driver对象属性

- | VolumeName 磁盘的卷标
- | DriverLetter 磁盘代号
- | SerialNumber 磁盘序列号
- | DriveType 磁盘种类
  - u 1--移动磁盘 2--本地磁盘 3--网络磁盘 4--光驱 5--其他
- | FileSystem 磁盘使用的文件系统
  - u 如FAT、FAT32、NTFS等
- | TotalSize 磁盘的使用空间
- | FreeSpace 磁盘的可用空间

## u 程序示范: 获取磁盘信息

```
function ShowDriveInfo(drvName)
{ //显示磁盘信息, 参数为磁盘名
  var fso, drv, s = "";
  fso = new ActiveXObject("Scripting.FileSystemObject");
  drv = fso.GetDrive(drvName);
  s = "磁盘 " + drvName + " - " + drv.VolumeName + "<br>";
  s += "磁盘序列号: " + drv.SerialNumber + " <br>";
  s += "可用空间: " + drv.TotalSize / 1024 + " Kb<br>";
  s += "剩余空间: " + drv.FreeSpace / 1024 + " Kb<br>";
  document.write(s);
}
ShowDriveInfo("C");
```



### 常用文件夹操作 (fso--FileSystemObject)

- ┆ 创建文件夹 **fso.CreateFolder**
- ┆ 删除文件夹 **Folder.Delete** 或 **fso.DeleteFolder**
- ┆ 移动文件夹 **Folder.Move** 或 **fso.MoveFolder**
- ┆ 复制文件夹 **Folder.Copy** 或 **fso.CopyFolder**
- ┆ 获取文件夹的名字 **Folder.Name**
- ┆ 检查文件夹是否存在 **fso.FolderExists**
- ┆ 获得现有 Folder 对象的实例 **fso.GetFolder**
- ┆ 查找父文件夹名字 **fso.GetParentFolderName**
- ┆ 找出系统文件夹的路径 **fso.GetSpecialFolder**

北京: 010-62196102 上海: 61202630 229 外企的师资、外企的技术、外企的品质

### 程序示范: 获取文件夹信息

```
var fso, fldr, s = "";
// 获得 FileSystemObject 的实例
fso = new ActiveXObject("Scripting.FileSystemObject");
fso.CreateFolder ("C:\\javascript"); // 创建新文件夹
document.write("已经创建文件夹 C:\\javascript" + "<br>");

fldr = fso.GetFolder("c:\\javascript"); // 获得 Drive 对象
pflr= fso.GetParentFolderName(fldr); //获得上一级路径名
document.write("父文件夹: " + pflr+ "<br>");
document.write("所在驱动盘 " + fldr.Drive + "<br>");

fso.DeleteFolder ("C:\\javascript"); // 删除新创建的文件夹
document.write("已删除文件夹 C:\\javascript" + "<br>");
```



北京: 010-62196102 上海: 61202630 230 外企的师资、外企的技术、外企的品质

### u 文件夹浏览

- l 浏览当前文件夹下所有子目录
  - u **folder.SubFolders** 子目录对象集合
- l 浏览当前文件夹下所有文件
  - u **folder.Files** 文件夹下所有文件对象集合
- l 遍历文件夹办法和遍历**Drivers**对象的方法类似
  - u 可以通过 **Enumerator** 对象和 **for** 语句进行遍历

### u 程序示范: 遍历文件夹

```
function ShowFolderList(folderspec){
    var fso, f, fc, s;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    f = fso.GetFolder(folderspec);
    fc = new Enumerator(f.SubFolders); //遍历子目录
    // fc = new Enumerator(f.Files); //遍历文件
    s = "";
    for (; !fc.atEnd(); fc.moveNext()) {
        s += fc.item();
        s += "<br>";
    }
    return(s);
}
```



### 常用文件操作 (fso--FileSystemObject)

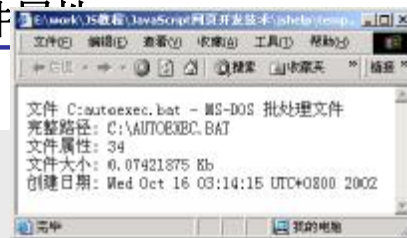
- 获取文件属性
- 创建与打开文件
- 读写文件内容
- 移动、复制和删除文件

### 文件对象属性(File)

| 属性               | 说明            |
|------------------|---------------|
| Name             | 文件名           |
| Type             | 文件类型          |
| Path             | 完整文件路径        |
| Attributes       | 文件属性, 如只读、系统等 |
| DateCreated      | 创建日期          |
| DateLastAccessed | 最后访问日期        |
| DateLastModified | 最后修改日期        |
| Size             | 文件大小          |

## 程序示范：获取文件信息

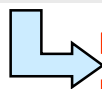
```
function ShowFileInfo(fileName)
{ //显示文件信息，参数为文件名
  var fso, fs, s = "";
  fso = new ActiveXObject("Scripting.FileSystemObject");
  fs = fso.GetFile(fileName);
  s = "文件 " + fileName + " - " + fs.Type + "<br>";
  s += "完整路径: " + fs.Path + " <br>";
  s += "文件属性: " + fs.Attributes + " <br>";
  s += "文件大小: " + fs.Size / 1024 + " Kb<br>";
  s += "创建日期: " + fs.DateCreated + " <br>";
  document.write(s);
}
ShowFileInfo("C:\\autoexec.bat");
```



## 创建与打开文件

- fso.CreateTextFile(strFilePath, flag)**  
 flag为true/false, 表示是否覆盖已有文件
- fso.OpenTextFile(strFilePath, flag1, flag2)**  
 flag1为数字, 1只读打开, 2可写打开  
 flag2为true/false, 不存在是否创建新文件

```
var fso, f1;
fso = new ActiveXObject("Scripting.FileSystemObject");
f1 = fso.CreateTextFile("c:\\testfile.txt", true); //新建文件
```



```
f1 = fso.OpenTextFile("c:\\testfile.txt", 2, true); //打开文件
```

## u 读取文件 ( fo=File Object )

- | **fo.Read(charNum)** 从文件当前位置读取charNum个字符
- | **fo.ReadAll()** 从文件对象fo读取全部内容
- | **fo.ReadLine()** 从文件中读取一行 (不包括换行符)
- | **fo.Skip(charNum)** 读取文件fo时跳过charNum个字符
- | **fo.SkipLine()** 文件当前位置跳到下一行

## u 写入文件

- | **fo.Write(string)** 向打开的文件写入字符串string
- | **fo.WriteBlankLines(lineNum)** 写入lineNum个换行符
- | **fo.WriteLine( [string] )** 写入字符串string再加上换行符

## u 程序示范: 读写文件内容

```
var fso, fs, s = "";  
fso = new ActiveXObject("Scripting.FileSystemObject");  
fo = fso.CreateTextFile("C:\\test.txt", true);  
fo.Write("hello");  
fo.WriteBlankLines(3);  
fo.WriteLine("How are you");  
fo.Close();  
fo = fso.OpenTextFile("C:\\test.txt", 1, true);  
document.write("读取一行: " + fo.ReadLine() + "<br>");  
document.write("读取其余全部: " + fo.ReadAll() + "<br>");  
fo.Close();
```



## 复制文件

### **fso.CopyFile ( source, destination[, overwrite] )**

- ü source 指定文件名字符串，允许使用通配符
- ü destination 目标地址(文件或文件夹)，不允许使用通配符
- ü overwrite 如果目标地址存在source指定文件，是否覆盖

```
fso = new ActiveXObject("Scripting.FileSystemObject");  
fso.CreateFolder("c:\\test\\");  
fso.CopyFile ("c:\\*.txt", "c:\\test\\");  
//复制C盘根目录下的所有文本文件到新建的C:\\test目录下
```

## 移动文件

### **fso.MoveFile ( source, destination )**

- ü source 指定文件名字符串，末尾允许使用通配符
- ü destination 目标地址，为文件夹路径

```
function MoveFile2C(filespec)  
{ //将文件移动到C盘根目录  
  var fso;  
  fso = new ActiveXObject("Scripting.FileSystemObject");  
  fso.MoveFile(filespec, "c:\\");  
}
```



## 删除文件

### fso.DeleteFile ( filespec[, force] )

- filespec 指定文件名字符串，末尾允许使用通配符
- force 为true/false，表示是否强制删除只读文件

```
function DeleteFile(filespec)
{ //删除指定文件
  var fso;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  fso.DeleteFile(filespec);
}
```

## Office对象介绍

- Microsoft Office提供了Automation接口来操作内部的应用程序对象、文档对象和工具栏对象等，这些对象统称Office对象
- JavaScript可以通过ActiveXObject对象和应用程序入口参数对Office对象进行操作

## Office操作对象的主要功能

- 可以操纵Office中的Word进行自动化文字处理
- 可以操纵Excel进行表格及报表处理
- 可以操纵PowerPoint进行幻灯片处理

## JavaScript中的Office操作

创建Office对象基本格式如下：

var appObject = new  
ActiveXObject(appName.appObject);

其中appName为字符串格式

| Office程序   | appName.appObject      |
|------------|------------------------|
| Word       | Word.Application       |
| Excel      | Excel.Application      |
| PowerPoint | Powerpoint.Application |

北京: 010-62196102 上海: 61202630 243 外企的师资、外企的技术、外企的品质

## 获取Office程序的版本



```
var appExcel = new ActiveXObject("Excel.Application");  
var appWord = new ActiveXObject("Word.Application");  
var appPowerpoint = new ActiveXObject("Powerpoint.Application");  
  
document.write('Excel版本 : ' + appExcel.Version);  
document.write('<br>Word版本 : ' + appWord.Version);  
document.write('<br>Powerpoint版本 : ' + appPowerpoint.Version);
```

通过类似的方式可以调用客户端  
的Office进行更多操作

北京: 010-62196102 上海: 61202630 244 外企的师资、外企的技术、外企的品质

## u 报表制作

- 由HTML表格输出为Excel表格



```
...
//获取需要导出的HTML内容,report1为一个Table
var pasteText = document.all.report1.innerHTML;
//复制文本内容到剪贴板
window.clipboardData.setData ("Text", pasteText);
// 创建Excel访问对象
var oXL = new ActiveXObject("Excel.Application");
oXL.Visible = true;
//新建工作簿
var oWB = oXL.Workbooks.Add();
var oSheet = oWB.ActiveSheet;
oSheet.Paste();
oXL.Visible = true;
oXL.UserControl = true;
...
```

北京: 010-62196102 上海: 61202630 245 外企的师资、外企的技术、外企的品质

## u 自定义对象介绍

- 对象是一种复合数据类型
- 自定义对象是指由自己定义属性和方法的对象
- 自定义对象可以扩充JavaScript的功能
- 自定义对象可以继承并扩充内置对象

## u JavaScript自定义对象的特点

- 可以通过扩展基础对象Object定义对象
- 可以通过构造函数直接定义对象
- 对象属性可以用数组方式访问
- JavaScript是Prototype-Based语言, 该机制使对象应用更简单

北京: 010-62196102 上海: 61202630 246 外企的师资、外企的技术、外企的品质

创建自定义对象：Object定义法

```
//使用Object创建自定义对象
var myHello=new Object();
myHello.s="cici";
myHello.hello=SayHello;
//对象方法具体实现
function SayHello(){
    var str="您好！现在是"+(new Date()).toLocaleString();
    window.status=this.s+str;
}
//调用对象方法
myHello.hello();
```

创建自定义对象：构造函数法

```
//使用构造函数创建自定义对象
function WebHello(s){
    this.s=s; //问候姓名
    this.hello=SayHello;
}
function SayHello(){
    var str="您好！现在是"+(new Date()).toLocaleString();
    window.status=this.s+str;
}
var myHello=new WebHello("cici");
myHello.hello();
```

### 使用Prototype属性扩展内置对象

- 例如要为 **Array** 对象添加返回数组中最大元素值的方法
- 实现方法定义一个获得最大值的函数，通过 **Array.prototype** 加入到内置对象，然后像普通方法一样使用它



//定义获取最大值的函数

```
function array_max(){  
    var i, max = this[0];  
    for (i = 1; i < this.length; i++){  
        if (max < this[i])  
            max = this[i];  
    }  
    return max;  
}
```

//加入Array对象原型

```
Array.prototype.max = array_max;  
var x = new Array(1, 2, 3, 4, 5, 6);  
alert( x.max() ); //返回6
```

### 组件制作介绍

- IE5.x版本开始提供DHTML Behaviors功能
- Behaviors是CSS样式中的动作样式定义
- Behaviors让HTML标签可以具有新的方法、事件和属性
- Behaviors将JavaScript独立存储在扩展名为.htc的文件
- HTML组件即HTML Components，缩写为HTC

### 组件带来的好处

- JavaScript代码和HTML更加独立，易于HTML编写
- 代码重用性增加
- 组件方便部署，易于维护，运行更加稳定

#### 创建组件

- 组件技术将原来的一个嵌入JavaScript代码的HTML页面分成两个文件，一个是调用Behaviors样式的HTML文件，一个是独立出来存放JavaScript程序和组件相关定义的HTC文件

```
<style>
.myedit{ behavior:url(test5-17.htc); }
</style>

...
选择您喜欢的城市(可以编辑)
<select style="width:90px;"class="myedit">
  <option>北京</option>
  <option>上海</option>
  <option>广州</option>
</select>
```

Behaviors的调用和其他样式一样

#### 可编辑下拉框组件HTC文件页面

```
<PUBLIC:COMPONENT>
<PUBLIC:ATTACH EVENT="oncontentready" ONEVENT="initBox()" />
<PUBLIC:ATTACH EVENT="onchange" ONEVENT="chgEdit()" />
<script language="JavaScript">
...
function initBox(){
...
}
function chgEdit(){
...
}
</script>
</PUBLIC:COMPONENT>
```

HTC组件采用XML格式编写，中间嵌入JavaScript代码

JavaScript作为一种应用广泛的Web编程语言，随着浏览器和网络的发展也在不断更新和成熟

以下提供的是一些优秀的JavaScript资源网站链接：

- | <http://webfx.eae.net/> 高级JavaScript特效制作技巧
- | <http://www.bindows.net/> 基于JavaScript和面向对象UI框架
- | <http://www.stedy.com/> 大量XP风格Web组件



北京：010-62196102 上海：61202630 253 外企的师资、外企的技术、外企的品质