# JavaScript学习

## 1、为什么学习JavaScript

### 一、学习的价值

1. 所有主流浏览器都支持JavaScript。

2. 目前，全世界大部分网页都使用JavaScript。

3. 它可以让网页呈现各种动态效果。

4. 做为一个Web开发师，如果你想提供漂亮的网页、令用户满意的上网体验，JavaScript是必不可少的工具。

### 二、易学行

1.学习环境无外不在，只要有文本编辑器，就能编写JavaScript程序。

2.我们可以用简单命令，完成一些基本操作。

### 三、从哪开始学习呢？

学习JavaScript的起点就是处理网页，所以我们先学习基础语法和如何使用DOM进行简单操作。

## 2、JavaScript应该放在哪里

### 一、放入html文件中

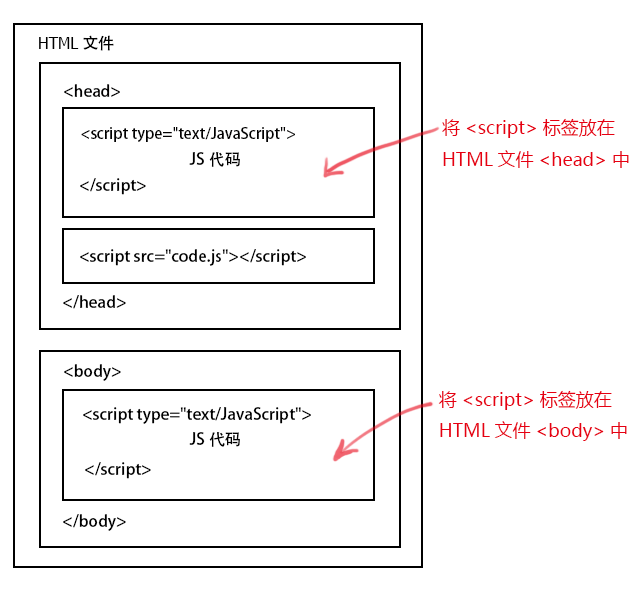
我们可以将JavaScript代码放在html文件中任何位置，但是我们一般放在网页的head或者body部分。

**放在<head>部分**

最常用的方式是在页面中head部分放置<script>元素，浏览器解析head部分就会执行这个代码，然后才解析页面的其余部分。

**放在<body>部分**

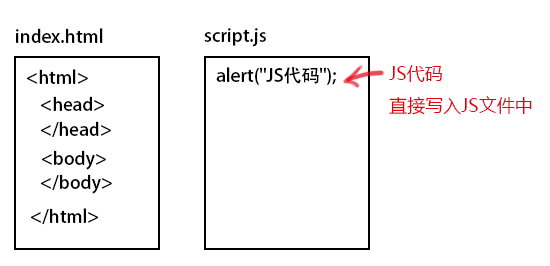
JavaScript代码在网页读取到该语句的时候就会执行。



注意: javascript作为一种脚本语言可以放在html页面中任何位置，但是浏览器解释html时是按先后顺序的，所以前面的script就先被执行。比如进行页面显示初始化的js必须放在head里面，因为初始化都要求提前进行（如给页面body设置css等）；而如果是通过事件调用执行的function那么对位置没什么要求的。

### 二、引用外部文件

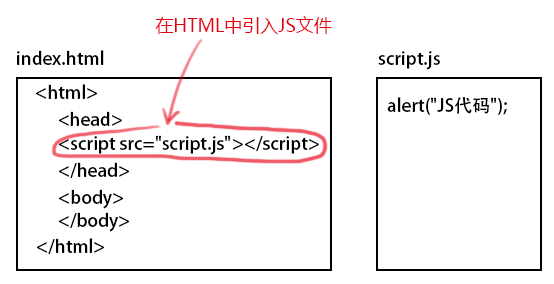
JavaScript代码只能写在HTML文件中吗?当然不是，我们可以把HTML文件和JS代码分开,并单独创建一个JavaScript文件(简称JS文件),其文件后缀通常为.js，然后将JS代码直接写在JS文件中。



注意:在JS文件中，不需要<script>标签,直接编写JavaScript代码就可以了。

JS文件不能直接运行，需嵌入到HTML文件中执行，我们需在HTML中添加如下代码，就可将JS文件嵌入HTML文件中。

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



## 3、JavaScript基础中的基础

### 一、JavaScript-认识语句和符号

JavaScript语句是发给浏览器的命令。这些命令的作用是告诉浏览器要做的事情。

每一句JavaScript代码格式: 语句;

先来看看下面代码

<script type="text/javascript">

alert("hello!");

</script>

例子中的alert("hello!");就是一个JavaScript语句。

一行的结束就被认定为语句的结束，通常在结尾加上一个分号";"来表示语句的结束。

看看下面这段代码,有三条语句，每句结束后都有";"，按顺序执行语句。

<script type="text/javascript">

document.write("I");

document.write("love");

document.write("JavaScript");

</script>

注意:

1. “;”分号要在英文状态下输入，同样，JS中的代码和符号都要在英文状态下输入。

2. 虽然分号“;”也可以不写，但我们要养成编程的好习惯，记得在语句末尾写上分号。

### 二、注释

注释的作用是提高代码的可读性，帮助自己和别人阅读和理解你所编写的JavaScript代码，注释的内容不会在网页中显示。注释可分为单行注释与多行注释两种。

我们为了方便阅读，注释内容一般放到需要解释语句的结尾处或周围。

单行注释，在注释内容前加符号 “//”。

<script type="text/javascript">

document.write("单行注释使用'//'"); // 我是注释，该语句功能在网页中输出内容

</script>

多行注释以"/\*"开始，以"\*/"结束。

<script type="text/javascript">

document.write("多行注释使用/\*注释内容\*/");

/\*

多行注释

养成书写注释的良好习惯

\*/

</script>

### 三、变量和操作符

#### （1）、变量

什么是变量? 从字面上看，变量是可变的量；从编程角度讲，变量是用于存储某种/某些数值的存储器。我们可以把变量看做一个盒子，为了区分盒子，可以用BOX1,BOX2等名称代表不同盒子，BOX1就是盒子的名字（也就是变量的名字）。

定义变量使用关键字var,语法如下：

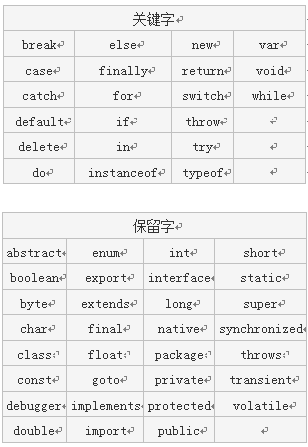
var 变量名

**变量名可以任意取名，但要遵循命名规则:**

1.变量必须使用字母、下划线(\_)或者美元符($)开始。

2.然后可以使用任意多个英文字母、数字、下划线(\_)或者美元符($)组成。

3.不能使用JavaScript关键词与JavaScript保留字。



变量要先声明再赋值，如下：

var mychar;

mychar="javascript";

var mynum = 6;

变量可以重复赋值，如下：

var mychar;

mychar="javascript";

mychar="hello";

#### （2）、操作符

操作符是用于在JavaScript中指定一定动作的符号。

1）操作符

看下面这段JavaScript代码。

sum = numa + numb;

其中的"="和"+"都是操作符。

JavaScript中还有很多这样的操作符，例如，算术操作符(+、-、\*、/等)，比较操作符(<、>、>=、<=等)，逻辑操作符(&&、||、！)。

注意: “=” 操作符是赋值，不是等于。

2) "+"操作符

算术运算符主要用来完成类似加减乘除的工作，在JavaScript中，“+”不只代表加法，还可以连接两个字符串，例如：

mystring = "Java" + "Script"; // mystring的值“JavaScript”这个字符串

3）、自加一，自减一 ( ++和- -)

算术操作符除了(+、-、\*、/)外，还有两个非常常用的操作符，自加一“++”；自减一“--”。首先来看一个例子：

mynum = 10;

mynum++; //mynum的值变为11

mynum--; //mynum的值又变回10

上面的例子中，mynum++使mynum值在原基础上增加1，mynum--使mynum在原基础上减去1,其实也可以写成:

mynum = mynum + 1;//等同于mynum++

mynum = mynum - 1;//等同于mynum--

4）、比较操作符

例子：

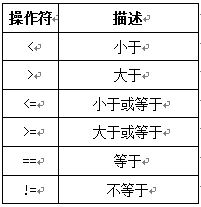
我们先来做道数学题，数学考试成绩中，小明考了90分，小红考了95分，问谁考的分数高?

答: 因为“95 > 90”，所以小红考试成绩高。

其中大于号">" 就是比较操作符，小红考试成绩和小明考试成绩就是操作数，并且是两个操作数。

也就是说两个操作数通过比较操作符进行比较，得到值为真（true）和假(false)。

在JavaScript中，这样的比较操作符有很多，这些操作符的含义如下:



看看下面例子:

var a = 5;//定义a变量，赋值为5

var b = 9; //定义b变量，赋值为9

document.write (a<b); //a小于b的值吗? 结果是真(true)

document.write (a>=b); //a大于或等于b的值吗? 结果是假(false)

document.write (a!=b); //a不等于b的值吗? 结果是真(true)

document.write (a==b); //a等于b的值吗? 结果是假(false)

5）、逻辑与操作符

**1）、与 &&**

数学里面的“a>b”，在JavaScript中还表示为a>b；数学中的“b大于a，b小于c”是“a<b<c”，那么在JavaScript中可以用&&表示，如下：

b>a && b<c //“&&”是并且的意思, 读法"b大于a"并且" b小于c "

**2）、|| 或**

"||"逻辑或操作符，相当于生活中的“或者”，当两个条件中有任一个条件满足，“逻辑或”的运算结果就为“真”。

例如：本周我们计划出游,可是周一至周五工作,所以周六或者周日哪天去都可以。即两天中只要有一天有空，就可以出游了。

var a=3;

var b=5;

var c;

c=b>a ||a>b; //b>a是true，a>b是false，c是true

**3）、！ 非**

"!"是逻辑非操作符，也就是"不是"的意思,非真即假，非假即真。好比小华今天买了一个杯子，小明说:"杯子是白色的"，小亮说:“杯子是红色的”，小华说："小明说的不是真话，小亮说的不是假话"。猜猜小华买的什么颜色的杯子，答案：红色杯子。

例如：

var a=3;

var b=5;

var c;

c=!(b>a); // b>a值是true,! (b>a）值是false

c=!(b<a); // b<a值是false, ! (b<a）值是true

（6）、操作符优先级

操作符之间的优先级（高到低）:

算术操作符 → 比较操作符 → 逻辑操作符 → "="赋值符号

如果同级的运算是按从左到右次序进行,多层括号由里向外。

var numa=3;

var numb=6;

jq= numa + 30 >10 && numb \* 3<2; //结果为false

### 四、流程控制语句

#### （1）、if..else语句

**注意：if小写，大写字母（IF）会出错！**

if...else语句是在指定的条件成立时执行代码，在条件不成立时执行else后的代码。

语法:

if(条件)

{ 条件成立时执行的代码 }

else

{ 条件不成立时执行的代码 }

假设我们通过年龄来判断是否为成年人，如年龄大于等于18岁，是成年人，否则不是成年人。代码表示如下:

<script type="text/javascript">

var myage = 18;

if(myage>=18) //myage>=18是判断条件

{ document.write("你是成年人。");}

else //否则年龄小于18

{ document.write("未满18岁，你不是成年人。");}

</script>

多重判断：

语法:

if(条件1)

{ 条件1成立时执行的代码}

else if(条件2)

{ 条件2成立时执行的代码}

...

else if(条件n)

{ 条件n成立时执行的代码}

else

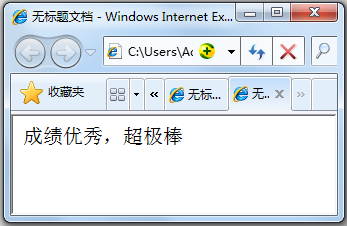
{ 条件1、2至n不成立时执行的代码}

例子：假设数学考试，小明考了86分，给他做个评价，60分以下的不及格，60(包含60分)-75分为良好，75(包含75分)-85分为很好，85(包含85分)-100优秀。

代码表示如下:



结果:



#### （2）、Switch语句

当有很多种选项的时候，switch比if else使用更方便。

语法:

switch(表达式)

{

case值1:

执行代码块 1

break;

case值2:

执行代码块 2

break;

...

case值n:

执行代码块 n

break;

default:

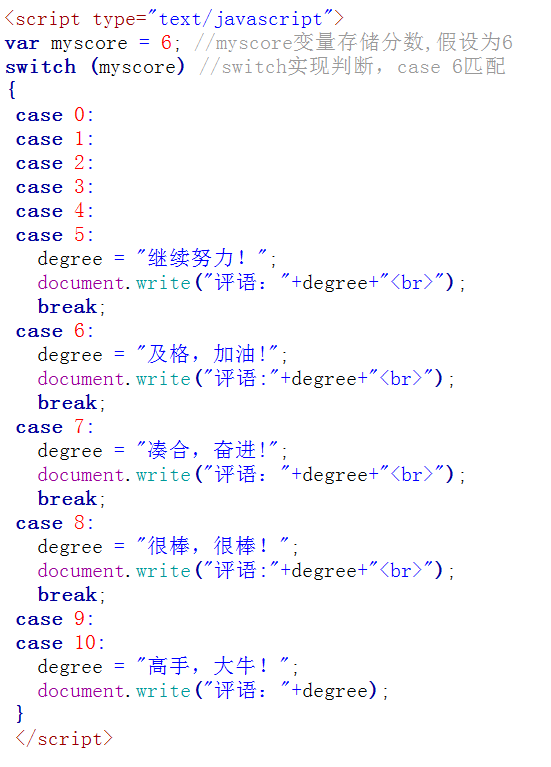
与 case值1 、 case值2...case值n 不同时执行的代码

}

语法说明:

Switch必须赋初始值，值与每个case值匹配。满足执行该 case 后的所有语句，并用break语句来阻止运行下一个case。如所有case值都不匹配，执行default后的语句。

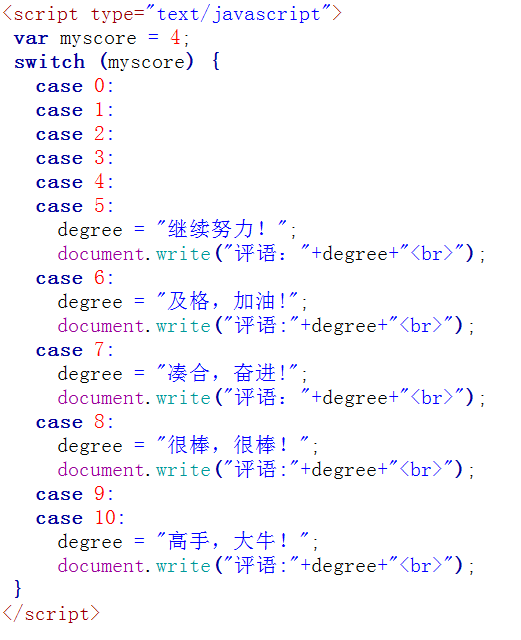
假设评价学生的考试成绩，10分满分制，我们按照每一分一个等级将成绩分等，并根据成绩的等级做出不同的评价



执行结果:

评语: 及格，加油!

注意:记得在case所执行的语句后添加上一个break语句。否则就直接继续执行下面的case中的语句，看以下代码:



**执行结果:**

评语: 继续努力!

评语: 及格，加油!

评语: 凑合，奋进

评语: 很棒，很棒

评语: 高手，大牛

在上面的代码中，没有break停止语句，如果成绩是4分，则case 5后面的语句将会得到执行，同样，case6、7-10后面的语句都会得到执行。

#### （3）、for循环

很多事情不只是做一次，要重复做。如打印10份试卷，每次打印一份，重复这个动作，直到打印完成。这些事情，我们使用循环语句来完成，循环语句，就是重复执行一段代码。

for语句结构：

for(初始化变量;循环条件;循环迭代)

{

循环语句

}

假如，一个盒子里有6个球，我们每次取一个，重复从盒中取出球，直到球取完为止。

<script type="text/javascript">

var num=1;

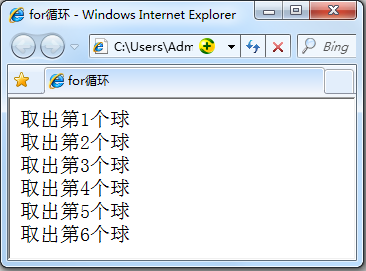
for (num=1;num<=6;num++) //初始化值；循环条件；循环后条件值更新

{ document.write("取出第"+num+"个球<br />");

}

</script>

结果:



执行思路:



#### （4）、while循环

和for循环有相同功能的还有while循环, while循环重复执行一段代码，直到某个条件不再满足。

**while语句结构：**

while(判断条件)

{

循环语句

}

使用while循环，完成从盒子里取球的动作，每次取一个，共6个球。

<script type="text/javascript">

var num=0; //初始化值

while (num<=6) //条件判断

{

document.write("取出第"+num+"个球<br />");

num=num+1; //条件值更新

}

</script>

#### （5）、Do...while循环

do while结构的基本原理和while结构是基本相同的，但是它保证循环体至少被执行一次。因为它是先执行代码，后判断条件，如果条件为真，继续循环。

do...while语句结构：

do

{

循环语句

}

while(判断条件)

我们试着输出5个数字。

<script type="text/javascript">

num= 1;

do

{

document.write("数值为:" + num+"<br />");

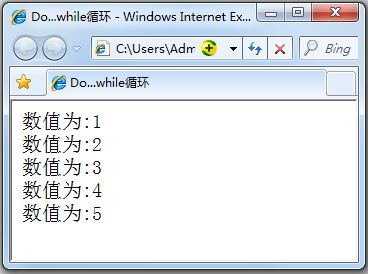
num++; //更新条件

}

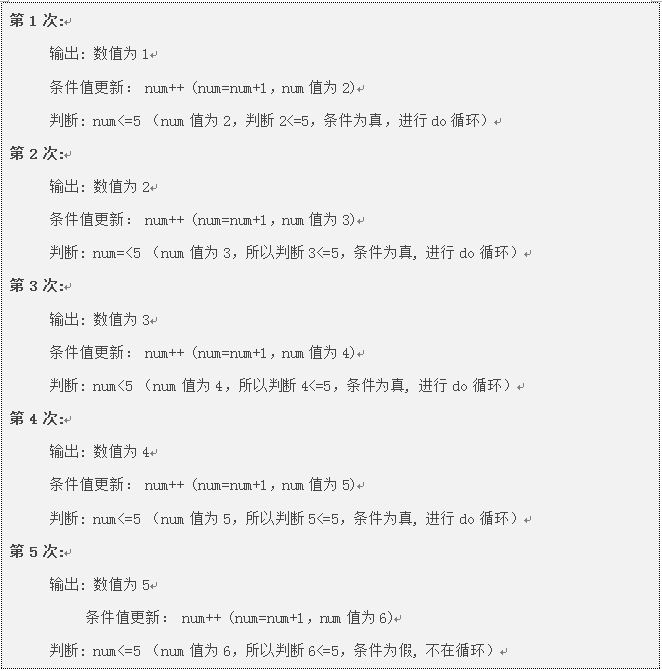
while (num<=5)

</script>

**执行结果:**



为什么呢?我们来看下执行思路：



#### （6）、退出循环break

在while、for、do...while、while循环中使用break语句退出当前循环，直接执行后面的代码。

**格式如下：**

for(初始条件;判断条件;循环后条件值更新)

{

if(特殊情况)

{break;}

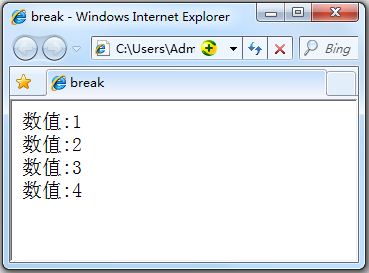
循环代码

}

当遇到特殊情况的时候，循环就会立即结束。看看下面的例子，输出10个数，如果数值为5，就停止输出。



**执行结果:**



#### （7）、继续循环continue

continue的作用是仅仅跳过本次循环，而整个循环体继续执行。

**语句结构：**

for(初始条件;判断条件;循环后条件值更新)

{

if(特殊情况)

{ continue; }

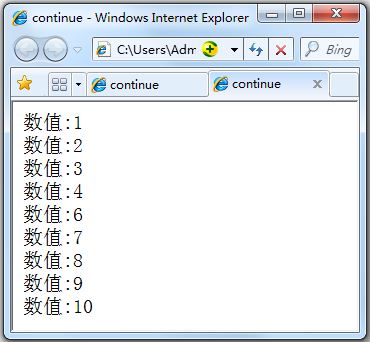
循环代码

}

上面的循环中，当特殊情况发生的时候，本次循环将被跳过，而后续的循环则不会受到影响。好比输出10个数字，如果数字为5就不输出了。



执行结果:



注:上面的代码中，num=5的那次循环将被跳过。

### 五、函数

是完成某个特定功能的一组语句。如没有函数，完成任务可能需要五行、十行、甚至更多的代码。这时我们就可以把完成特定功能的代码块放到一个函数里，直接调用这个函数，就省重复输入大量代码的麻烦。

如何定义一个函数呢？基本语法如下:

function 函数名()

{

函数代码;

}

说明:

1. function定义函数的关键字。

2. "函数名"你为函数取的名字。

3. "函数代码"替换为完成特定功能的代码。

我们来编写一个实现两数相加的简单函数,并给函数起个有意义的名字：“add2”，代码如下：

function add2(){

var sum = 3 + 2;

alert(sum);

}

函数调用:

函数定义好后，是不能自动执行的，所以需调用它,只需直接在需要的位置写函数就ok了,代码如下:



**返回值：**

思考:上一节函数中，通过"document.write"把结果输出来，如果想对函数的结果进行处理怎么办呢？

我们只要把"document.write(sum)"这行改成如下代码：

function add2(x,y)

{

sum = x + y;

return sum; //返回函数值,return后面的值叫做返回值。

}

还可以通过变量存储调用函数的返回值，代码如下:

result = add2(3,4);//语句执行后,result变量中的值为7。

**注意:函数中参数和返回值不只是数字，还可以是字符串等其它类型。**

### 六、打印输出

document.write() 可用于直接向 HTML 输出流写内容。简单的说就是直接在网页中输出内容。

第一种:输出内容用""括起，直接输出""号内的内容。

<script type="text/javascript">

document.write("I love JavaScript！"); //内容用""括起来，""里的内容直接输出。

</script>

第二种:通过变量，输出内容

<script type="text/javascript">

var mystr="hello world!";

document.write(mystr); //直接写变量名，输出变量存储的内容。

</script>

第三种:输出多项内容，内容之间用+号连接。

<script type="text/javascript">

var mystr="hello";

document.write(mystr+"I love JavaScript"); //多项内容之间用+号连接

第四种:输出HTML标签，并起作用，标签使用""括起来。

<script type="text/javascript">

var mystr="hello";

document.write(mystr+"<br>");//输出hello后，输出一个换行符

document.write("JavaScript");

</script>

第五种 输出空格

在写JS代码的时候，大家可以会发现这样现象:

document.write(" 1 2 3 ");

结果: 1 2 3

无论在输出的内容中什么位置有多少个空格，显示的结果好像只有一个空格。

这是因为浏览器显示机制，对手动敲入的空格，将连续多个空格显示成1个空格。

解决方法:

1. 使用输出html标签&nbsp;来解决

document.write("&nbsp;&nbsp;"+"1"+"&nbsp;&nbsp;&nbsp;&nbsp;"+"23");

结果: 1 23

2. 使用CSS样式来解决

document.write("<span style='white-space:pre;'>"+" 1 2 3 "+"</span>");

结果: 1 2 3

在输出时添加“white-space:pre;”样式属性。这个样式表示"空白会被浏览器保留"

### 七、DOM操作

**1、认识DOM**

文档对象模型DOM（Document Object Model）定义访问和处理HTML文档的标准方法。DOM 将HTML文档呈现为带有元素、属性和文本的树结构（节点树）。

先来看看下面代码:



将HTML代码分解为DOM节点层次图:



HTML文档可以说由节点构成的集合，三种常见的DOM节点:

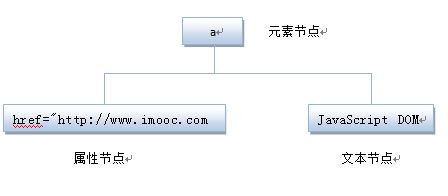
1. 元素节点：上图中<html>、<body>、<p>等都是元素节点，即标签。

2. 文本节点:向用户展示的内容，如<li>...</li>中的JavaScript、DOM、CSS等文本。

3. 属性节点:元素属性，如<a>标签的链接属性href="http://www.imooc.com"。

看下面代码:

<a href="http://www.imooc.com">JavaScript DOM</a>



**2、通过ID获取元素**

学过HTML/CSS样式，都知道，网页由标签将信息组织起来，而标签的id属性值是唯一的，就像是每人有一个身份证号一样，只要通过身份证号就可以找到相对应的人。那么在网页中，我们通过id先找到标签，然后进行操作。

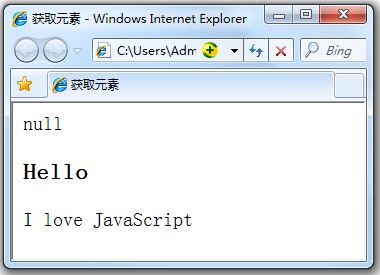
语法:

document.getElementById(“id”)

看看下面代码:



结果:null或[object HTMLParagraphElement]



注:获取的元素是一个对象，如想对元素进行操作，我们要通过它的属性或方法。

**3、innerHTML 属性**

innerHTML 属性用于获取或替换 HTML 元素的内容。

语法:

Object.innerHTML

注意:

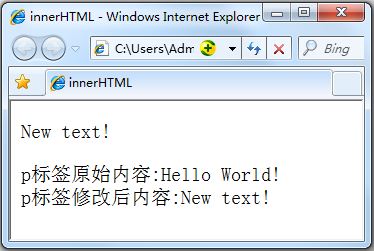
1.Object是获取的元素对象，如通过document.getElementById("ID")获取的元素。

2.注意书写，innerHTML区分大小写。

我们通过id="con"获取<p> 元素，并将元素的内容输出和改变元素内容，代码如下:



结果:



**4、改变 HTML 样式**

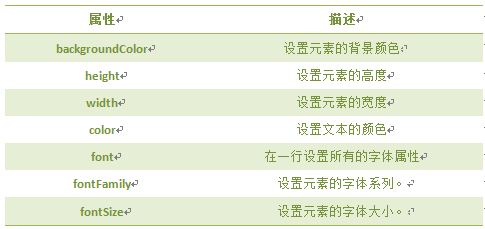
HTML DOM 允许 JavaScript 改变 HTML 元素的样式。如何改变 HTML 元素的样式呢？

语法:

Object.style.property=new style;

注意:Object是获取的元素对象，如通过document.getElementById("id")获取的元素。

基本属性表（property）:



注意:该表只是一小部分CSS样式属性，其它样式也可以通过该方法设置和修改。

看看下面的代码:

改变 <p> 元素的样式，将颜色改为红色，字号改为20,背景颜色改为蓝：

<p id="pcon">Hello World!</p>

<script>

var mychar = document.getElementById("pcon");

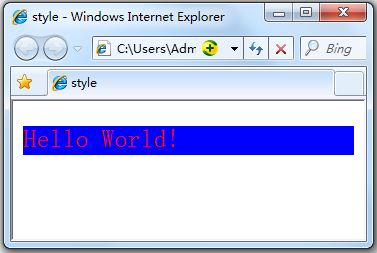
mychar.style.color="red";

mychar.style.fontSize="20";

mychar.style.backgroundColor ="blue";

</script>

结果:



**5、显示和隐藏（display属性）**

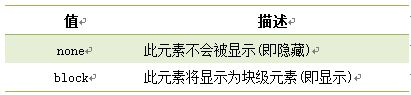
网页中经常会看到显示和隐藏的效果，可通过display属性来设置。

语法：

Object.style.display = value

注意:Object是获取的元素对象，如通过document.getElementById("id")获取的元素。

value取值:



看看下面代码:



**6、控制类名（className 属性）**

className 属性设置或返回元素的class 属性。

语法：

object.className = classname

作用:

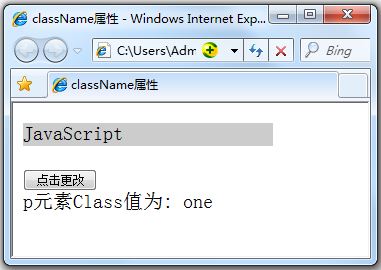
1.获取元素的class 属性

2. 为网页内的某个元素指定一个css样式来更改该元素的外观

看看下面代码，获得 <p> 元素的 class 属性和改变className：



结果:



### 八、事件

#### （1）、什么是事件：

JavaScript 创建动态页面。事件是可以被 JavaScript 侦测到的行为。 网页中的每个元素都可以产生某些可以触发 JavaScript 函数或程序的事件。

比如说，当用户单击按钮或者提交表单数据时，就发生一个鼠标单击（onclick）事件，需要浏览器做出处理，返回给用户一个结果。

主要事件表:



#### （2）、鼠标单击事件( onclick ）

onclick是鼠标单击事件，当在网页上单击鼠标时，就会发生该事件。同时onclick事件调用的程序块就会被执行，通常与按钮一起使用。

比如，我们单击按钮时，触发 onclick 事件，并调用两个数和的函数add2()。代码如下：

<html>

<head>

<script type="text/javascript">

function add2(){

var numa,numb,sum;

numa=6;

numb=8;

sum=numa+numb;

document.write("两数和为:"+sum); }

</script>

</head>

<body>

<form>

<input name="button" type="button" value="点击提交" onclick="add2()" />

</form>

</body>

</html>

**注意: 在网页中，如使用事件，就在该元素中设置事件属性。**

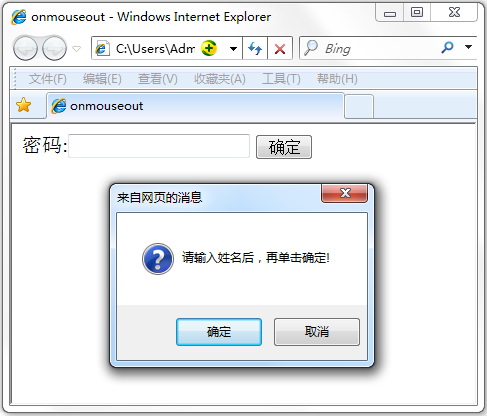
#### （3）、鼠标经过事件（onmouseover）

鼠标经过事件，当鼠标移到一个对象上时，该对象就触发onmouseover事件，并执行onmouseover事件调用的程序。

现实鼠标经过"确定"按钮时，触发onmouseover事件，调用函数info()，弹出消息框，代码如下:



运行结果:



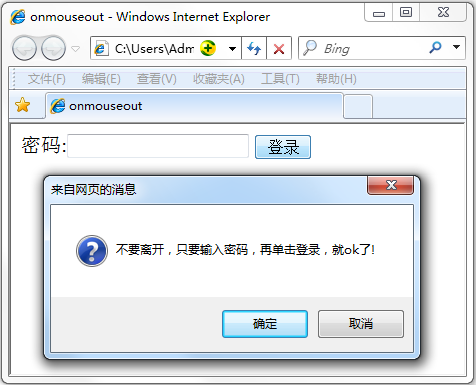
#### （4）、鼠标移开事件

鼠标移开事件，当鼠标移开当前对象时，执行onmouseout调用的程序。

当把鼠标移动到"登录"按钮上，然后再移开时，触发onmouseout事件，调用函数message()，代码如下:



运行结果:



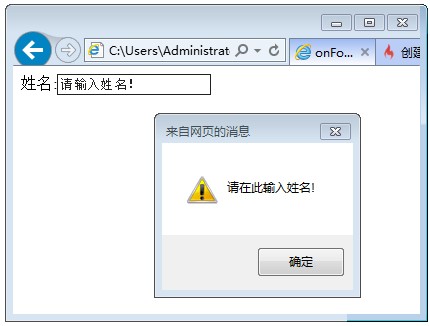
#### （5）、光标聚焦事件（onfocus）

当网页中的对象获得聚点时，执行onfocus调用的程序就会被执行。

如下代码, 当将光标移到文本框内时，即焦点在文本框内，触发onfocus 事件，并调用函数message()。



运行结果：



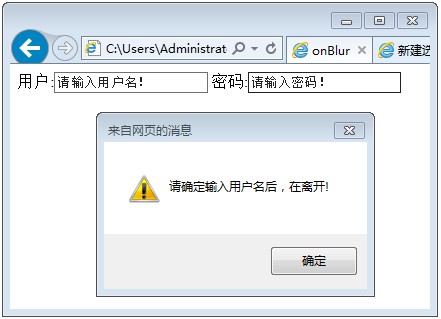
#### （6）、失焦事件（onblur）

onblur事件与onfocus是相对事件，当光标离开当前获得聚焦对象的时候，触发onblur事件，同时执行被调用的程序。

如下代码, 网页中有用户和密码两个文本框。当前光标在用户文本框内时（即焦点在文本框），在光标离开该文本框后（即失焦时），触发onblur事件，并调用函数message()。



运行结果：



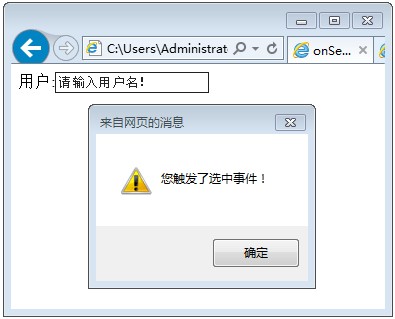
#### （7）、内容选中事件（onselect）

选中事件，当文本框或者文本域中的文字被选中时，触发onselect事件，同时调用的程序就会被执行。

如下代码,当选中用户文本框内的文字时，触发onselect 事件，并调用函数message()。



运行结果：



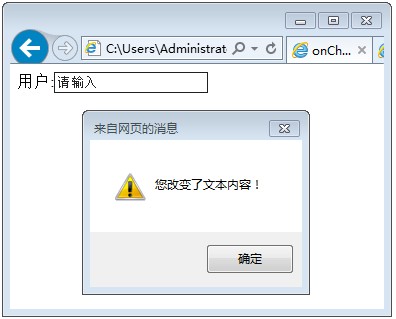
#### （8）、文本框内容改变事件（onchange）

通过改变文本框的内容来触发onchange事件，同时执行被调用的程序。

如下代码,当用户将文本框内的文字改变后，弹出对话框“您改变了文本内容！”。



运行结果：



#### （9）、加载事件（onload）

事件会在页面加载完成后，立即发生，同时执行被调用的程序。

注意：

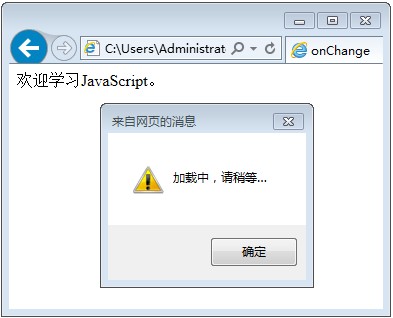
1. 加载页面时，触发onload事件，事件写在<body>标签内。

2. 此节的加载页面，可理解为打开一个新页面时。

如下代码,当加载一个新页面时，弹出对话框“加载中，请稍等…”。



运行结果：



#### （10）、卸载事件（onunload）

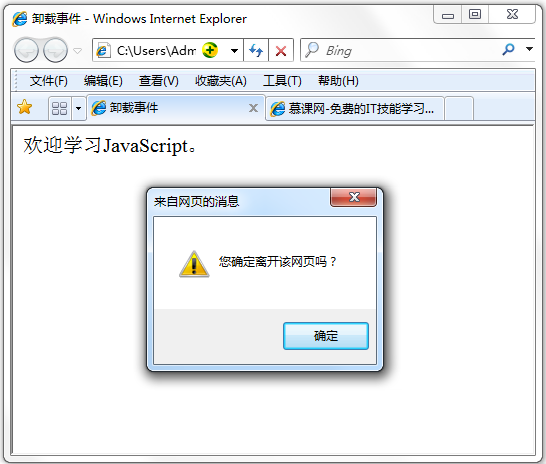
当用户退出页面时（页面关闭、页面刷新等），触发onUnload事件，同时执行被调用的程序。

注意：不同浏览器对onunload事件支持不同。

如下代码,当退出页面时，弹出对话框“您确定离开该网页吗？”



运行结果：（IE浏览器）



#### （11）、编程练习

写一个简单的计算器：实现2个输入框中输入整数后，点击第三个输入框能给出2个整数的加减乘除。

思路：

第一步： 创建构建运算函数count()。

第二步： 获取两个输入框中的值和获取选择框的值。

提示：document.getElementById( id名 ).value 获取或设置 id名的值。

第三步： 获取通过下拉框来选择的值来改变加减乘除的运算法则。

提示：使用switch判断运算法则。

第四步： 通过 = 按钮来调用创建的函数，得到结果。

注意: 使用parseInt()函数可解析一个字符串,并返回一个整数。

<!DOCTYPE html>

<html>

<head>

<title> 事件</title>

<script type="text/javascript">

function count(){

//获取第一个输入框的值

//获取第二个输入框的值

//获取选择框的值

//获取通过下拉框来选择的值来改变加减乘除的运算法则

//设置结果输入框的值

var txt1=document.getElementById("txt1").value;

var select=document.getElementById("select").value;

var txt2=document.getElementById("txt2").value;

switch(select)

{

case "+":

document.getElementById("fruit").value=parseFloat(txt1)+parseFloat(txt2);

break;

case "-":

document.getElementById("fruit").value=parseFloat(txt1)-parseFloat(txt2);

break;

case "\*":

document.getElementById("fruit").value=parseFloat(txt1)\*parseFloat(txt2);

break;

case "/":

document.getElementById("fruit").value=parseFloat(txt1)/parseFloat(txt2);

break;

}

}

</script>

</head>

<body>

<input type='text' id='txt1' />

<select id='select'>

<option value='+'>+</option>

<option value="-">-</option>

<option value="\*">\*</option>

<option value="/">/</option>

</select>

<input type='text' id='txt2' />

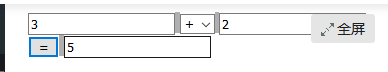
<input type='button' value=' = ' onclick="count()" /> <!--通过 = 按钮来调用创建的函数，得到结果-->

<input type='text' id='fruit' />

</body>

</html>

运行结果：



### 九、JavaScript内置对象

#### **（1）、什么是对象**

JavaScript 中的所有事物都是对象，如:字符串、数值、数组、函数等，每个对象带有属性和方法。

对象的属性：反映该对象某些特定的性质的，如：字符串的长度、图像的长宽等；

对象的方法：能够在对象上执行的动作。例如，表单的“提交”(Submit)，时间的“获取”(getYear)等；

JavaScript 提供多个内建对象，比如 String、Date、Array 等等，使用对象前先定义，如下使用数组对象：

var objectName =new Array();//使用new关键字定义对象

或者

var objectName =[];

访问对象属性的语法:

objectName.propertyName

如使用 Array 对象的 length 属性来获得数组的长度：

var myarray=new Array(6);//定义数组对象

var myl=myarray.length;//访问数组长度length属性

以上代码执行后，myl的值将是：6

访问对象的方法：

objectName.methodName()

如使用string 对象的 toUpperCase() 方法来将文本转换为大写：

var mystr="Hello world!";//创建一个字符串

var request=mystr.toUpperCase(); //使用字符串对象方法

以上代码执行后，request的值是：HELLO WORLD!

#### **（2）、Date 日期对象**

日期对象可以储存任意一个日期，并且可以精确到毫秒数（1/1000 秒）。

定义一个时间对象 :

var Udate=new Date();

注意:使用关键字new，Date()的首字母必须大写。

使 Udate 成为日期对象，并且已有初始值：当前时间(当前电脑系统时间)。

如果要自定义初始值，可以用以下方法：

var d = new Date(2012, 10, 1); //2012年10月1日

var d = new Date('Oct 1, 2012'); //2012年10月1日

我们最好使用下面介绍的“方法”来严格定义时间。

访问方法语法：“<日期对象>.<方法>”

Date对象中处理时间和日期的常用方法：



get/setDay() 返回/设置星期

**1）、返回/设置年份方法：**get/setFullYear()

get/setFullYear() 返回/设置年份，用四位数表示。

var mydate=new Date();//当前时间2014年3月6日

document.write(mydate+"<br>");//输出当前时间

document.write(mydate.getFullYear()+"<br>");//输出当前年份

mydate.setFullYear(81); //设置年份

document.write(mydate+"<br>"); //输出年份被设定为 0081年。

注意:不同浏览器， mydate.setFullYear(81)结果不同，年份被设定为 0081或81两种情况。

结果:

Thu Mar 06 2014 10:57:47 GMT+0800

2014

Thu Mar 06 0081 10:57:47 GMT+0800

注意:

1.结果格式依次为：星期、月、日、年、时、分、秒、时区。(火狐浏览器)

2. 不同浏览器，时间格式有差异。

**2）、返回星期方法（试过设置，但设置不成功）**getDay()

getDay() 返回星期，返回的是0-6的数字，0 表示星期天。如果要返回相对应“星期”，通过数组完成，代码如下:

<script type="text/javascript">

var mydate=new Date();//定义日期对象

var weekday=["星期日","星期一","星期二","星期三","星期四","星期五","星期六"];

//定义数组对象,给每个数组项赋值

var mynum=mydate.getDay();//返回值存储在变量mynum中

document.write(mydate.getDay());//输出getDay()获取值

document.write("今天是："+ weekday[mynum]);//输出星期几

</script>

注意：以上代码是在2014年3月7日，星期五运行。

结果:

5

今天是：星期五

**3）、返回/设置时间方法**

get/setTime() 返回/设置时间，单位毫秒数，**计算从 1970 年 1 月 1 日零时到日期对象所指的日期的毫秒数**。

如果将目前日期对象的时间推迟1小时，代码如下:

<script type="text/javascript">

var mydate=new Date();

document.write("当前时间："+mydate+"<br>");

mydate.setTime(mydate.getTime() + 60 \* 60 \* 1000);

document.write("推迟一小时时间：" + mydate);

</script>

结果:

当前时间：Thu Mar 6 11:46:27 UTC+0800 2014

推迟一小时时间：Thu Mar 6 12:46:27 UTC+0800 2014

注意: 1. 一小时 60 分，一分 60 秒，一秒 1000 毫秒

2. 时间推迟 1 小时,就是: “x.setTime(x.getTime() + 60 \* 60 \* 1000);”

#### **（3）、String 字符串对象**

在之前的学习中已经使用字符串对象了，定义字符串的方法就是直接赋值。比如：

var mystr = "I love JavaScript!"

定义mystr字符串后，我们就可以访问它的属性和方法。

访问字符串对象的属性length:

stringObject.length; 返回该字符串的长度。

var mystr="Hello World!";

var myl=mystr.length;

以上代码执行后，myl 的值将是：12

访问字符串对象的方法：

使用 String 对象的 toUpperCase() 方法来将字符串小写字母转换为大写：

var mystr="Hello world!";

var mynum=mystr.toUpperCase();

以上代码执行后，mynum 的值是：HELLO WORLD!

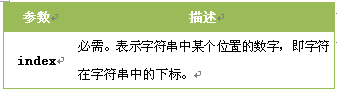
**1）、返回指定位置的字符：**

charAt() 方法可返回指定位置的字符。返回的字符是长度为 1 的字符串。

语法:

stringObject.charAt(index)

参数说明：



注意：

1.字符串中第一个字符的下标是 0。最后一个字符的下标为字符串长度减一（string.length-1）。

2.如果参数 index 不在 0 与 string.length-1 之间，该方法将返回一个空字符串。

如:在字符串 "I love JavaScript!" 中，返回位置2的字符：

<script type="text/javascript">

var mystr="I love JavaScript!"

document.write(mystr.charAt(2));

</script>

注意：一个空格也算一个字符。

以上代码的运行结果：

I

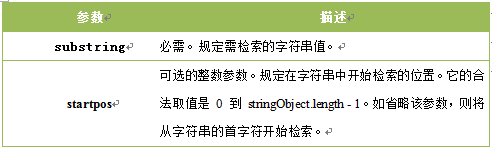
**2）、返回指定的字符串首次出现的位置**

ndexOf() 方法可返回某个指定的**字符串**值在字符串中首次出现的位置。

语法

stringObject.indexOf(substring, startpos)

参数说明：



说明：

1.该方法将从头到尾地检索字符串 stringObject，看它是否含有子串 substring。

2.可选参数，从stringObject的startpos位置开始查找substring，如果没有此参数将从stringObject的开始位置查找。

3.如果找到一个 substring，则返回 substring 的第一次出现的位置。stringObject 中的字符位置是从 0 开始的。

注意：

1.indexOf() 方法区分大小写。

2.如果要检索的字符串值没有出现，则该方法返回 -1。

例如: 对 "I love JavaScript!" 字符串内进行不同的检索：

<script type="text/javascript">

var str="I love JavaScript!"

document.write(str.indexOf("I") + "<br />");

document.write(str.indexOf("v") + "<br />");

document.write(str.indexOf("v",8));

</script>

以上代码的输出：

0

4

9

**3）、字符串分割split()**

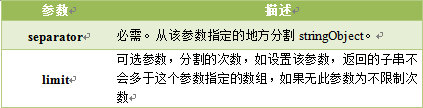
知识讲解：

split() 方法将字符串分割为字符串数组，并返回此数组。

语法：

stringObject.split(separator,limit)

参数说明:



注意：如果把空字符串 ("") 用作 separator，那么 stringObject 中的每个字符之间都会被分割。

我们将按照不同的方式来分割字符串：

使用指定符号分割字符串，代码如下:

var mystr = "www.imooc.com";

document.write(mystr.split(".")+"<br>");

document.write(mystr.split(".", 2)+"<br>");

运行结果:

www,imooc,com

www,imooc

将字符串分割为字符，代码如下：

document.write(mystr.split("")+"<br>");

document.write(mystr.split("", 5));

运行结果:

w,w,w,.,i,m,o,o,c,.,c,o,m

w,w,w,.,i

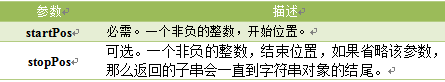
**4）、提取字符串substring()**

substring() 方法用于提取字符串中介于两个指定下标之间的字符。

语法:

stringObject.substring(startPos,stopPos)

参数说明:



注意：

1. 返回的内容是从 start开始(包含start位置的字符)到 stop-1 处的所有字符，其长度为 stop 减start。

2. 如果参数 start 与 stop 相等，那么该方法返回的就是一个空串（即长度为 0 的字符串）。

3. 如果 start 比 stop 大，那么该方法在提取子串之前会先交换这两个参数。

使用 substring() 从字符串中提取字符串，代码如下：

<script type="text/javascript">

var mystr="I love JavaScript";

document.write(mystr.substring(7));

document.write(mystr.substring(2,6));

</script>

运行结果:

JavaScript

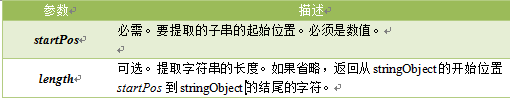
love

**5）、substr() 方法从字符串中提取从 startPos位置开始的指定数目的字符串。**

语法:

stringObject.substr(startPos,length)

参数说明:



注意：

1、如果参数startPos是负数，从字符串的尾部开始算起的位置。也就是说，-1 指字符串中最后一个字符，-2 指倒数两个字符，以此类推，可以不用设置length。

2、如果startPos为负数且绝对值大于字符串长度，startPos为0。

使用 substr() 从字符串中提取一些字符，代码如下：

<script type="text/javascript">

var mystr="I love JavaScript!";

document.write(mystr.substr(7));

document.write(mystr.substr(2,4));

</script>

运行结果：

JavaScript!

love

#### **（4）、Math对象**

Math对象，提供对数据的数学计算。

使用 Math 的属性和方法，代码如下：

<script type="text/javascript">

var mypi=Math.PI;

var myabs=Math.abs(-15);

document.write(mypi);

document.write(myabs);

</script>

运行结果:

3.141592653589793

15

注意：Math 对象是一个固有的对象，无需创建它，直接把 Math 作为对象使用就可以调用其所有属性和方法。这是它与Date,String对象的区别。

Math 对象属性



Math 对象方法



**1）、向上取整ceil()**

ceil() 方法可对一个数进行向上取整。

语法:

Math.ceil(x)

参数说明:

http://img.mukewang.com/532fe8e00001e4e605230063.jpg

注意：它返回的是大于或等于x，并且与x最接近的整数。

我们将把 ceil() 方法运用到不同的数字上，代码如下：

<script type="text/javascript">

document.write(Math.ceil(0.8) + "<br />")

document.write(Math.ceil(6.3) + "<br />")

document.write(Math.ceil(5) + "<br />")

document.write(Math.ceil(3.5) + "<br />")

document.write(Math.ceil(-5.1) + "<br />")

document.write(Math.ceil(-5.9))

</script>

运行结果：

1

7

5

4

-5

-5

**2）、向下取整floor()**

floor() 方法可对一个数进行向下取整。

语法:

Math.floor(x)

参数说明：

http://img.mukewang.com/532fe8e00001e4e605230063.jpg

注意：返回的是小于或等于x，并且与 x 最接近的整数。

我们将在不同的数字上使用 floor() 方法，代码如下:

<script type="text/javascript">

document.write(Math.floor(0.8)+ "<br>")

document.write(Math.floor(6.3)+ "<br>")

document.write(Math.floor(5)+ "<br>")

document.write(Math.floor(3.5)+ "<br>")

document.write(Math.floor(-5.1)+ "<br>")

document.write(Math.floor(-5.9))

</script>

运行结果：

0

6

5

3

-6

-6

**3）、四舍五入round()**

round() 方法可把一个数字四舍五入为最接近的整数。

语法:

Math.round(x)

参数说明：

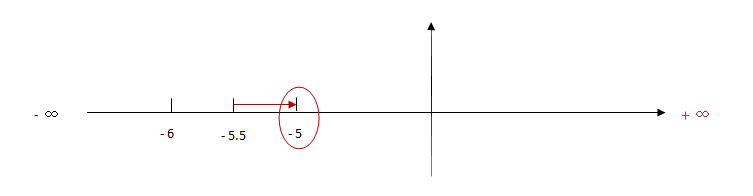
http://img.mukewang.com/532feb70000180ab03210059.jpg

注意：

1. 返回与 x 最接近的整数。

2. 对于 5.5，该方法将进行上舍入。(5.5 将舍入为 6)

3. **如果 x 与两侧整数同等接近，则结果接近 +∞方向的数字值 。(如 -5.5 将舍入为 -5; -5.52 将舍入为 -6),如下图:**



把不同的数舍入为最接近的整数,代码如下：

<script type="text/javascript">

document.write(Math.round(1.6)+ "<br>");

document.write(Math.round(2.5)+ "<br>");

document.write(Math.round(0.49)+ "<br>");

document.write(Math.round(-6.4)+ "<br>");

document.write(Math.round(-6.5)+ "<br>");

document.write(Math.round(-6.6));

</script>

运行结果：

2

3

0

-6

-6

-7

**4）、随机数 random()**

random() 方法可返回介于 0 ~ 1（**大于或等于 0 但小于 1** )之间的一个随机数。

语法：

Math.random();

注意：返回一个大于或等于 0 但小于 1 的符号为正的数字值。

我们取得介于 0 到 1 之间的一个随机数，代码如下：

<script type="text/javascript">

document.write(Math.random());

</script>

运行结果：

0.190305486195328

注意:因为是随机数，所以每次运行结果不一样，但是0 ~ 1的数值。

获得0 ~ 10之间的随机数，代码如下:

<script type="text/javascript">

document.write((Math.random())\*10);

</script>

运行结果：

8.72153625893887

获得0 ~ 10之间的**随机整数**，代码如下:

<script type="text/javascript">

document.write(Math.round(Math.random()\*10));

</script>

运行结果：

9

#### （5）、数组

数组对象是一个对象的集合，里边的对象可以是不同类型的。数组的每一个成员对象都有一个“下标”，用来表示它在数组中的位置，是从零开始的

**1）、创建数组**

【1】. 定义了一个空数组:

var 数组名= new Array();

【2】. 定义时指定有n个空元素的数组：

var 数组名 =new Array(n);

【3】.定义数组的时候，直接初始化数据：

var 数组名 = [<元素1>, <元素2>, <元素3>...];

如下：

var myArray = [2, 8, 6];

说明：定义了一个数组 myArray，里边的元素是：myArray[0] = 2; myArray[1] = 8; myArray[2] = 6。

注意：

1.创建的新数组是空数组，没有值，如输出，则显示undefined。

2.虽然创建数组时，指定了长度，但实际上数组都是变长的，也就是说**即使指定了长度为8，仍然可以将元素存储在规定长度以外。**

3. 数组存储的数据可以是任何类型（数字、字符、布尔值等）

**2）、数组赋值**

步骤：

第一步：创建数组var myarr=new Array();

第二步：给数组赋值

myarr[1]=" 张三";

myarr[2]=" 李四";

例如：

下面创建一个数组，用于存储5个人的数学成绩。

var myarray=new Array(); //创建一个新的空数组

myarray[0]=66; //存储第1个人的成绩

myarray[1]=80; //存储第2个人的成绩

myarray[2]=90; //存储第3个人的成绩

myarray[3]=77; //存储第4个人的成绩

myarray[4]=59; //存储第5个人的成绩

注意：数组每个值有一个索引号，从0开始。

**如何向数组增加一个新的元素呢？**

只需使用下一个未用的索引，任何时刻可以不断向数组增加新元素。

myarray[5]=88; //使用一个新索引，为数组增加一个新元素

**3）、二维数组**

一维数组，我们看成一组盒子，每个盒子只能放一个内容。

一维数组的表示: myarray[ ]

二维数组，我们看成一组盒子，不过每个盒子里还可以放多个盒子。

二维数组的表示: myarray[ ][ ]

注意: 二维数组的两个维度的索引值也是从0开始，两个维度的最后一个索引值为长度-1。

1. 二维数组的定义方法一

var myarr=new Array(); //先声明一维

for(var i=0;i<2;i++){ //一维长度为2

myarr[i]=new Array(); //再声明二维

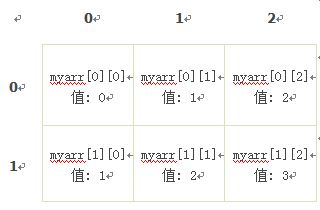
for(var j=0;j<3;j++){ //二维长度为3

myarr[i][j]=i+j; // 赋值，每个数组元素的值为i+j

}

}

将上面二维数组，用表格的方式表示:



2. 二维数组的定义方法二

var Myarr = [[0 , 1 , 2 ],[1 , 2 , 3]];

3. 赋值

myarr[0][1]=5; //将5的值传入到数组中，覆盖原有值。

说明: myarr[0][1] ,0 表示表的行，1表示表的列。

**4）、数组属性length**

如果我们想知道数组的大小，只需引用数组的一个属性length。Length属性表示数组的长度，即数组中元素的个数。

语法：

myarray.length; //获得数组myarray的长度

注意：因为数组的索引总是由0开始，所以一个数组的上下限分别是：0和length-1。如数组的长度是5，数组的上下限分别是0和4。

var arr=[55,32,5,90,60,98,76,54];//包含8个数值的数组arr

document.write(arr.length); //显示数组长度8

document.write(arr[7]); //显示第8个元素的值54

同时，JavaScript数组的length属性是可变的，这一点需要特别注意。

arr.length=10; //增大数组的长度

document.write(arr.length); //数组长度已经变为10

数组随元素的增加，长度也会改变，如下:

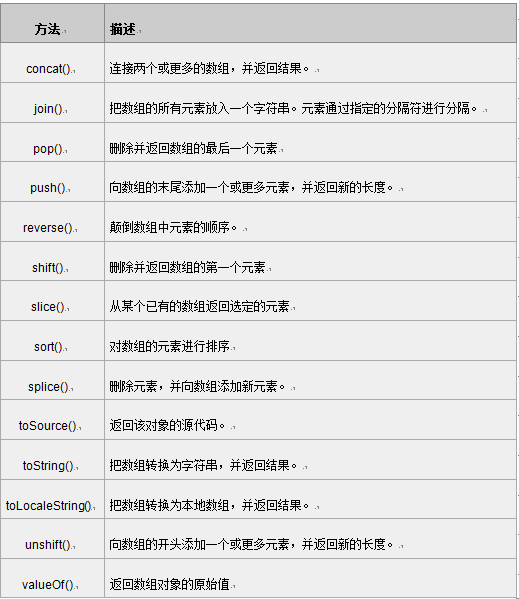
var arr=[98,76,54,56,76]; // 包含5个数值的数组

document.write(arr.length); //显示数组的长度5

arr[15]=34; //增加元素，使用索引为15,赋值为34

alert(arr.length); //显示数组的长度16 （也就中间的空值也算了）

数组还有其他方法如下：



**5）、数组连接concat()**

concat() 方法用于连接两个或多个数组。此方法返回一个新数组，不改变原来的数组。

语法

arrayObject.concat(array1,array2,...,arrayN)

参数说明：



注意: **该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。**

我们创建一个数组，将把 concat() 中的参数连接到数组 myarr 中，代码如下：

<script type="text/javascript">

var mya = new Array(3);

mya[0] = "1";

mya[1] = "2";

mya[2] = "3";

document.write(mya.concat(4,5)+"<br>");

document.write(mya);

</script>

运行结果：

1,2,3,4,5

1,2,3

我们创建了三个数组，然后使用 concat() 把它们连接起来，代码如下：

<script type="text/javascript">

var mya1= new Array("hello!")

var mya2= new Array("I","love");

var mya3= new Array("JavaScript","!");

var mya4=mya1.concat(mya2,mya3);

document.write(mya4);

</script>

运行结果：

hello!,I,love,JavaScript,!

**6）、指定分隔符连接数组元素join()**

join()方法用于把数组中的所有元素放入一个字符串。元素是通过指定的分隔符进行分隔的。

语法：

arrayObject.join(分隔符)

http://img.mukewang.com/533297ff0001516905150059.jpg

注意：返回一个字符串，该字符串把数组中的各个元素串起来，用<分隔符>置于元素与元素之间。这个方法不影响数组原本的内容。 我们使用join（）方法，将数组的所有元素放入一个字符串中，代码如下：

<script type="text/javascript">

var myarr = new Array(3);

myarr[0] = "I";

myarr[1] = "love";

myarr[2] = "JavaScript";

document.write(myarr.join());

</script>

运行结果：

I,love,JavaScript

我们将使用分隔符来分隔数组中的元素，代码如下：

<script type="text/javascript">

var myarr = new Array(3)

myarr[0] = "I";

myarr[1] = "love";

myarr[2] = "JavaScript";

document.write(myarr.join("."));

</script>

运行结果：

I.love.JavaScript

**7）、颠倒数组元素顺序reverse()**

reverse() 方法用于颠倒数组中元素的顺序。

语法：

arrayObject.reverse()

注意：**该方法会改变原来的数组，而不会创建新的数组**。

定义数组myarr并赋值，然后颠倒其元素的顺序：

<script type="text/javascript">

var myarr = new Array(3)

myarr[0] = "12"

myarr[1] = "23"

myarr[2] = "34"

document.write(myarr + "<br />")

document.write(myarr.reverse())

</script>

运行结果：

12,23,34

34,23,12

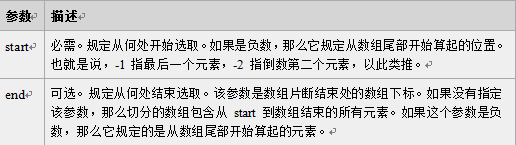
**（8）、选定元素slice()**

slice() 方法可从已有的数组中返回选定的元素。

语法

arrayObject.slice(start,end)

**参数说明：**



**1.返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。**

**2. 该方法并不会修改数组，而是返回一个子数组。**

注意：

1. 可使用负值从数组的尾部选取元素。

2.如果 end 未被规定，那么 slice() 方法会选取从 start 到数组结尾的所有元素。

3. String.slice() 与 Array.slice() 相似。

我们将创建一个新数组，然后从其中选取的元素，代码如下：

<script type="text/javascript">

var myarr = new Array(1,2,3,4,5,6);

document.write(myarr + "<br>");

document.write(myarr.slice(2,4) + "<br>");

document.write(myarr.slice(-2) + "<br>");

document.write(myarr);

</script>

运行结果：

1,2,3,4,5,6

3,4

5,6

1,2,3,4,5,6

**8)、数组排序sort()**

ort()方法使数组中的元素按照一定的顺序排列。

语法:

arrayObject.sort(方法函数)

http://img.mukewang.com/53329a2a000127f705170060.jpg

1.如果不指定<方法函数>，则按unicode码顺序排列。

2.如果指定<方法函数>，则按<方法函数>所指定的排序方法排序。

myArray.sort(sortMethod);

注意: 该函数要比较两个值，然后返回一个用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数 a 和 b，其返回值如下：

若返回值<=-1，则表示 A 在排序后的序列中出现在 B 之前。

若返回值>-1 && <1，则表示 A 和 B 具有相同的排序顺序。

若返回值>=1，则表示 A 在排序后的序列中出现在 B 之后。

1.使用sort()将数组进行排序，代码如下：

<script type="text/javascript">

var myarr1 = new Array("Hello","John","love","JavaScript");

var myarr2 = new Array("80","16","50","6","100","1");

document.write(myarr1.sort()+"<br>");

document.write(myarr2.sort());

</script>

运行结果：

Hello,JavaScript,John,love

1,100,16,50,6,80

注意:上面的代码没有按照数值的大小对数字进行排序。

2.如要实现这一点，就必须使用一个排序函数，代码如下：

<script type="text/javascript">

function sortNum(a,b) {

return a - b;

//升序，如降序，把“a - b”该成“b - a”

}

var myarr = new Array("80","16","50","6","100","1");

document.write(myarr + "<br>");

document.write(myarr.sort(sortNum));

</script>

运行结果：

80,16,50,6,100,1

1,6,16,50,80,100

#### （6）、window对象

window对象是BOM的核心，window对象指当前的浏览器窗口。

window对象方法:



**1）、警告（alert 消息对话框）**

我们在访问网站的时候，有时会突然弹出一个小窗口，上面写着一段提示信息文字。如果你不点击“确定”，就不能对网页做任何操作，这个小窗口就是使用alert实现的。

语法:

alert(字符串或变量);

看下面的代码:

<script type="text/javascript">

var mynum = 30;

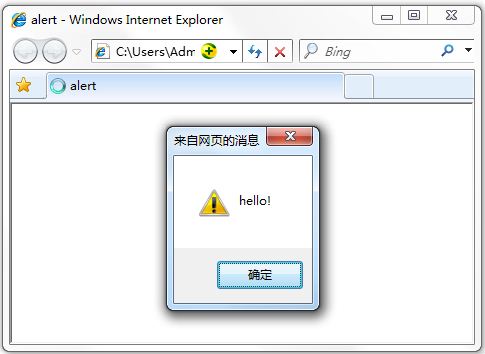
alert("hello!");

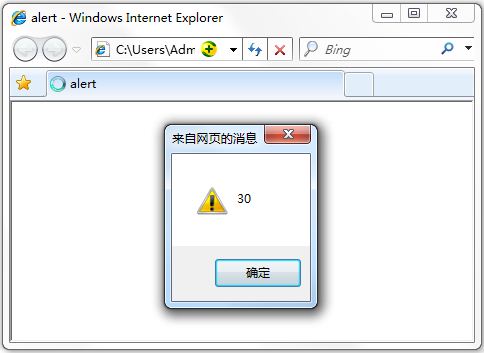
alert(mynum);

</script>

注:alert弹出消息对话框(包含一个确定按钮)。

结果:按顺序弹出消息框





注意:

1. 在点击对话框"确定"按钮前，不能进行任何其它操作。

2. 消息对话框通常可以用于调试程序。

3. alert输出内容，可以是字符串或变量，与document.write 相似。

**2）、确认（confirm 消息对话框）**

confirm 消息对话框通常用于允许用户做选择的动作，如：“你对吗？”等。弹出对话框(包括一个确定按钮和一个取消按钮)。

语法:

confirm(str);

参数说明:

str：在消息对话框中要显示的文本

返回值: Boolean值

返回值:

当用户点击"确定"按钮时，返回true

当用户点击"取消"按钮时，返回false

注: 通过返回值可以判断用户点击了什么按钮

看下面的代码:

<script type="text/javascript">

var mymessage=confirm("你喜欢JavaScript吗?");

if(mymessage==true)

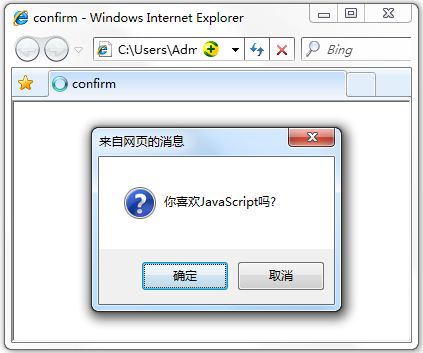
{ document.write("很好,加油!"); }

else

{ document.write("JS功能强大，要学习噢!"); }

</script>

结果:



**注: 消息对话框是排它的，即用户在点击对话框按钮前，不能进行任何其它操作。**

**3）、提问（prompt 消息对话框）**

prompt弹出消息对话框,通常用于询问一些需要与用户交互的信息。弹出消息对话框（包含一个确定按钮、取消按钮与一个文本输入框）。

语法:

prompt(str1, str2);

参数说明：

str1: 要显示在消息对话框中的文本，不可修改

str2：文本框中的内容，可以修改

返回值:

1. 点击确定按钮，文本框中的内容将作为函数返回值

2. 点击取消按钮，将返回null

看看下面代码:

var myname=prompt("请输入你的姓名:");

if(myname!=null)

{ alert("你好"+myname); }

else

{ alert("你好 my friend."); }

结果:

注:在用户点击对话框的按钮前，不能进行任何其它操作。

**4）、打开和关闭窗口（window.open）**

**打开窗口：**

open() 方法可以查找一个已经存在或者新建的浏览器窗口。

语法：

window.open([URL], [窗口名称], [参数字符串])

参数说明:

URL：可选参数，在窗口中要显示网页的网址或路径。如果省略这个参数，或者它的值是空字符串，那么窗口就不显示任何文档。

窗口名称：可选参数，被打开窗口的名称。

1.该名称由字母、数字和下划线字符组成。

2."\_top"、"\_blank"、"\_self"具有特殊意义的名称。

\_blank：在新窗口显示目标网页

\_self：在当前窗口显示目标网页

\_top：框架网页中在上部窗口中显示目标网页

3.相同 name 的窗口只能创建一个，要想创建多个窗口则 name 不能相同。

4.name 不能包含有空格。

参数字符串：可选参数，设置窗口参数，各参数用逗号隔开。

参数表:



例如:打开http://www.imooc.com网站，大小为300px \* 200px，无菜单，无工具栏，无状态栏，有滚动条窗口：

<script type="text/javascript"> window.open('http://www.imooc.com','\_blank','width=300,height=200,menubar=no,toolbar=no, status=no,scrollbars=yes')

</script>

注意：运行结果考虑浏览器兼容问题。

**关闭窗口：**

close()关闭窗口

用法：

window.close(); //关闭本窗口

或

<窗口对象>.close(); //关闭指定的窗口

例如:关闭新建的窗口。

<script type="text/javascript">

var mywin=window.open('http://www.imooc.com'); //将新打的窗口对象，存储在变量mywin中

mywin.close();

</script>

注意:上面代码在打开新窗口的同时，关闭该窗口，看不到被打开的窗口。

**5）、JavaScript 计时器**

在JavaScript中，我们可以在设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。

计时器类型：

一次性计时器：仅在指定的延迟时间之后触发一次。

间隔性触发计时器：每隔一定的时间间隔就触发一次。

计时器方法：



**计时器setInterval()**

在执行时,从载入页面后每隔指定的时间执行代码。

语法:

setInterval(代码,交互时间);

参数说明：

1. 代码：要调用的函数或要执行的代码串。

2. 交互时间：周期性执行或调用表达式之间的时间间隔，以毫秒计（1s=1000ms）。

返回值:

一个可以传递给 clearInterval() 从而取消对"代码"的周期性执行的值。

调用函数格式(假设有一个clock()函数):

setInterval("clock()",1000)

或

setInterval(clock,1000)

我们设置一个计时器，每隔100毫秒调用clock()函数，并将时间显示出来，代码如下:

<!DOCTYPE HTML>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>计时器</title>

<script type="text/javascript">

var int=setInterval(clock, 100)

function clock(){

var time=new Date();

document.getElementById("clock").value = time;

}

</script>

</head>

<body>

<form>

<input type="text" id="clock" size="50" />

</form>

</body>

</html>

**取消计时器clearInterval()**

clearInterval() 方法可取消由 setInterval() 设置的交互时间。

语法：

clearInterval(id\_of\_setInterval)

参数说明:

id\_of\_setInterval：由 setInterval() 返回的 ID 值。

每隔 100 毫秒调用 clock() 函数,并显示时间。当点击按钮时，停止时间,代码如下:

<!DOCTYPE HTML>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>计时器</title>

<script type="text/javascript">

function clock(){

var time=new Date();

document.getElementById("clock").value = time;

}

// 每隔100毫秒调用clock函数，并将返回值赋值给i

var i=setInterval("clock()",100);

</script>

</head>

<body>

<form>

<input type="text" id="clock" size="50" />

<input type="button" value="Stop" onclick="clearInterval(i)" />

</form>

</body>

</html>

**计时器setTimeout()**

setTimeout()计时器，在载入后延迟指定时间后,去执行一次表达式,仅执行一次。

语法:

setTimeout(代码,延迟时间);

参数说明：

1. 要调用的函数或要执行的代码串。

2. 延时时间：在执行代码前需等待的时间，以毫秒为单位（1s=1000ms)。

当我们打开网页3秒后，在弹出一个提示框，代码如下:

<!DOCTYPE HTML>

<html>

<head>

<script type="text/javascript">

setTimeout("alert('Hello!')", 3000 );

</script>

</head>

<body>

</body>

</html>

当按钮start被点击时，setTimeout()调用函数，在5秒后弹出一个提示框。

<!DOCTYPE HTML>

<html>

<head>

<script type="text/javascript">

function tinfo(){

var t=setTimeout("alert('Hello!')",5000);

}

</script>

</head>

<body>

<form>

<input type="button" value="start" onClick="tinfo()">

</form>

</body>

</html>

要创建一个运行于无穷循环中的计数器，我们需要编写一个函数来调用其自身。在下面的代码，当按钮被点击后，输入域便从0开始计数。

<!DOCTYPE HTML>

<html>

<head>

<script type="text/javascript">

var num=0;

function numCount(){

document.getElementById('txt').value=num;

num=num+1;

setTimeout("numCount()",1000);

}

</script>

</head>

<body>

<form>

<input type="text" id="txt" />

<input type="button" value="Start" onClick="numCount()" />

</form>

</body>

</html>

**取消计时器clearTimeout()**

setTimeout()和clearTimeout()一起使用，停止计时器。

语法:

clearTimeout(id\_of\_setTimeout)

参数说明:

id\_of\_setTimeout：由 setTimeout() 返回的 ID 值。该值标识要取消的延迟执行代码块。

下面的例子和上节的无穷循环的例子相似。唯一不同是，现在我们添加了一个 "Stop" 按钮来停止这个计数器：

<!DOCTYPE HTML>

<html>

<head>

<script type="text/javascript">

var num=0,i;

function timedCount(){

document.getElementById('txt').value=num;

num=num+1;

i=setTimeout(timedCount,1000);

}

setTimeout(timedCount,1000);

function stopCount(){

clearTimeout(i);

}

</script>

</head>

<body>

<form>

<input type="text" id="txt">

<input type="button" value="Stop" onClick="stopCount()">

</form>

</body>

</html>

**7）、History 对象**

history对象记录了用户曾经浏览过的页面(URL)，并可以实现浏览器前进与后退相似导航的功能。

注意:从窗口被打开的那一刻开始记录，每个浏览器窗口、每个标签页乃至每个框架，都有自己的history对象与特定的window对象关联。

语法：

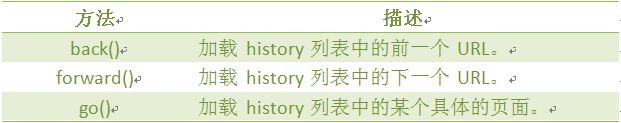
window.history.[属性|方法]

注意：window可以省略。

History 对象属性



History 对象方法



使用length属性，当前窗口的浏览历史总长度，代码如下：

<script type="text/javascript">

var HL = window.history.length;

document.write(HL);

</script>

**返回前一个浏览的页面**

back()方法，加载 history 列表中的前一个 URL。

语法：

window.history.back();

比如，返回前一个浏览的页面，代码如下：

window.history.back();

注意：等同于点击浏览器的倒退按钮。

back()相当于go(-1),代码如下:

window.history.go(-1);

**返回下一个浏览的页面**

forward()方法，加载 history 列表中的下一个 URL。

如果倒退之后，再想回到倒退之前浏览的页面，则可以使用forward()方法,代码如下:

window.history.forward();

注意：等价点击前进按钮。

forward()相当于go(1),代码如下:

window.history.go(1);

**返回浏览历史中的其他页面**

go()方法，根据当前所处的页面，加载 history 列表中的某个具体的页面。

语法：

window.history.go(number);

参数：



浏览器中，返回当前页面之前浏览过的第二个历史页面，代码如下：

window.history.go(-2);

注意：和在浏览器中单击两次后退按钮操作一样。

同理，返回当前页面之后浏览过的第三个历史页面，代码如下：

window.history.go(3);

**8）、Location对象**

location用于获取或设置窗体的URL，并且可以用于解析URL。

语法:

location.[属性|方法]

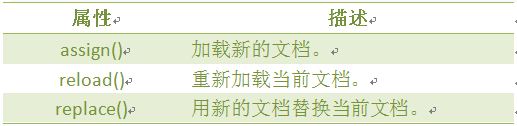
location对象属性图示:



location 对象属性：



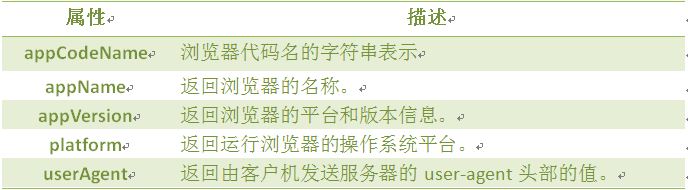
location 对象方法:



**9）、Navigator对象**

Navigator 对象包含有关浏览器的信息，通常用于检测浏览器与操作系统的版本。

对象属性:



查看浏览器的名称和版本，代码如下:

<script type="text/javascript">

var browser=navigator.appName;

var b\_version=navigator.appVersion;

document.write("Browser name"+browser);

document.write("<br>");

document.write("Browser version"+b\_version);

</script>

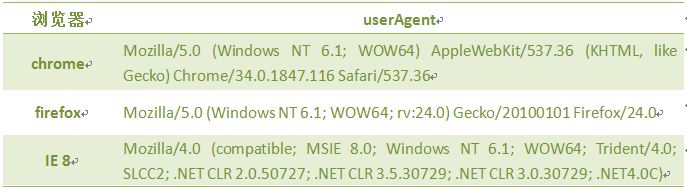
**userAgent**

返回用户代理头的字符串表示(就是包括浏览器版本信息等的字符串)

语法

navigator.userAgent

几种浏览的user\_agent.，像360的兼容模式用的是IE、极速模式用的是chrom的内核。



使用userAgent判断使用的是什么浏览器(假设使用的是IE8浏览器),代码如下:

function validB(){

var u\_agent = navigator.userAgent;

var B\_name="Failed to identify the browser";

if(u\_agent.indexOf("Firefox")>-1){

B\_name="Firefox";

}else if(u\_agent.indexOf("Chrome")>-1){

B\_name="Chrome";

}else if(u\_agent.indexOf("MSIE")>-1&&u\_agent.indexOf("Trident")>-1){

B\_name="IE(8-10)";

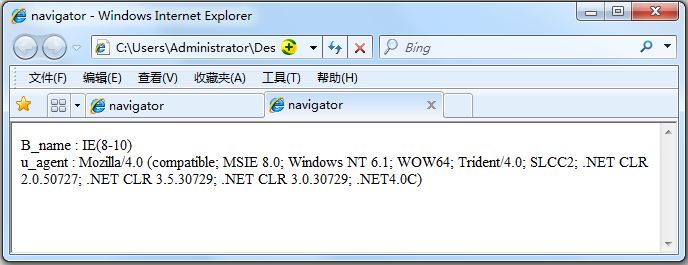
}

document.write("B\_name:"+B\_name+"<br>");

document.write("u\_agent:"+u\_agent+"<br>");

}

运行结果:



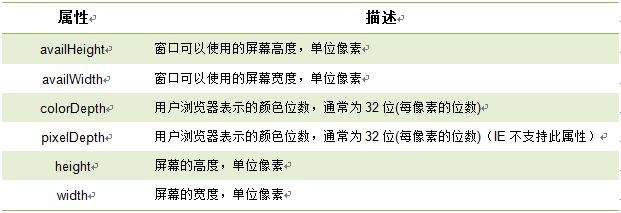
**10）、screen对象**

screen对象用于获取用户的屏幕信息。

语法：

window.screen.属性

对象属性:



**屏幕分辨率的高和宽**

window.screen 对象包含有关用户屏幕的信息。

1. screen.height 返回屏幕分辨率的高

2. screen.width 返回屏幕分辨率的宽

注意:

1.单位以像素计。

2. **window.screen 对象在编写时可以不使用 window 这个前缀。**

我们来获取屏幕的高和宽，代码如下:

<script type="text/javascript">

document.write( "屏幕宽度："+screen.width+"px<br />" );

document.write( "屏幕高度："+screen.height+"px<br />" );

</script>

**屏幕可用高和宽度**

1. screen.availWidth 属性返回访问者屏幕的宽度，以像素计，减去界面特性，比如任务栏。

2. screen.availHeight 属性返回访问者屏幕的高度，以像素计，减去界面特性，比如任务栏。

注意:

不同系统的任务栏默认高度不一样，及任务栏的位置可在屏幕上下左右任何位置，所以有可能可用宽度和高度不一样。

我们来获取屏幕的可用高和宽度，代码如下：

<script type="text/javascript">

document.write("可用宽度：" + screen.availWidth);

document.write("可用高度：" + screen.availHeight);

</script>

注意:根据屏幕的不同显示值不同。