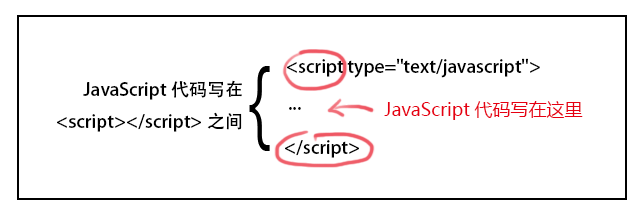
* 1. JavaScript

可以用文本编辑器编写JavaScript程序。

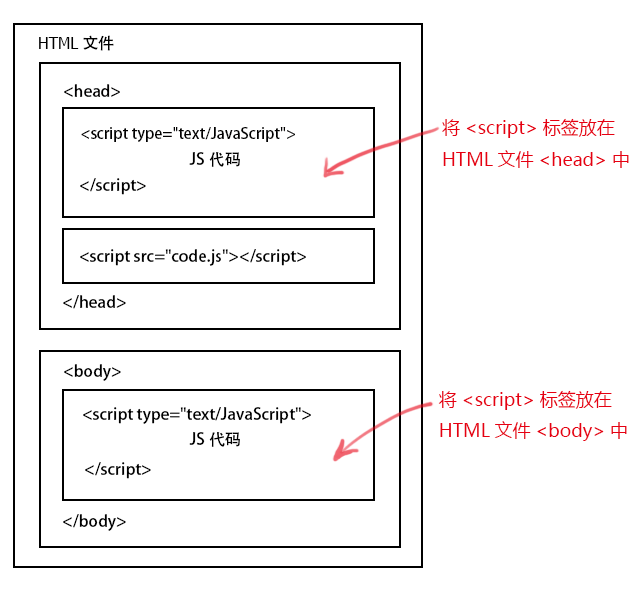
* 1. 如何插入JS（插入方法与HTML文件插入CSS样式一样）



1-3 引用JS外部文件

添加<script src="script.js"></script>代码将JS文件嵌入HTML文件中。

1-4 JS在页面中的位置



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。

1-6 注释

**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>

1-7 变量

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;

var num1,mun2 ; //声明多个变量

mychar="javascript";

var mynum = 6;

变量可以重复赋值，如下：

var mychar;

mychar="javascript";

mychar="hello";

注意:

1. 在JS中区分大小写，如变量mychar与myChar是不一样的，表示是两个变量。

2. 变量虽然也可以不声明，直接使用，但不规范，需要先声明，后使用。

1-8 if...else

1-9 函数

1. 基本语法 2、两数相加的简单函数

function 函数名() function add2(){

{ var sum = 3 + 2;

函数代码; alert(sum);

} }



2-1 JavaScript-输出内容（document.write）

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

</script>

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>

**JS中如何输出空格**

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**"

2-2 **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;

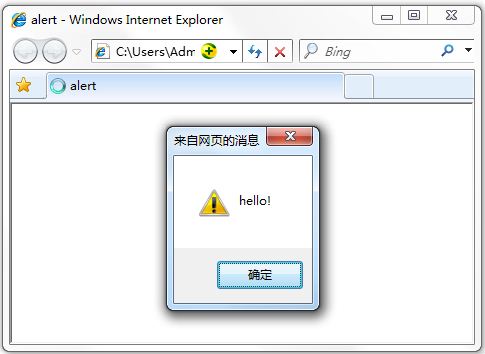
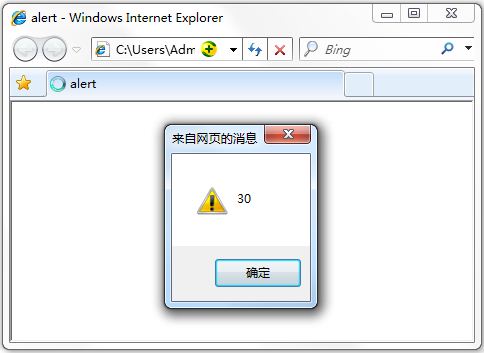
alert("hello!");

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**

**[](http://img.mukewang.com/52e362430001bdd204850354.jpg)** [](http://img.mukewang.com/52e362850001024d04840353.jpg)

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。

2-3 **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:** 看下面的代码:

**confirm(str);** <script type="text/javascript">

**参数说明:** var mymessage=confirm("你喜欢JavaScript吗?");

**str：**在消息对话框中要显示的文本 if(mymessage==true)

**返回值:** Boolean值 { document.write("很好,加油!"); }

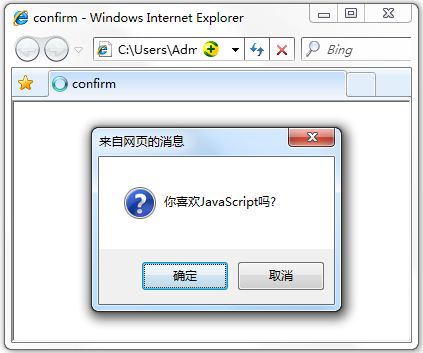
**返回值:** else

当用户点击"确定"按钮时，返回true { document.write("JS功能强大，要学习噢!"); }

当用户点击"取消"按钮时，返回false </script>

**注:** **通过返回值可以判断用户点击了什么按钮**

**结果:**

**[](http://img.mukewang.com/52e35bc60001f01a04230353.jpg)**

**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。**

2-4 **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:");

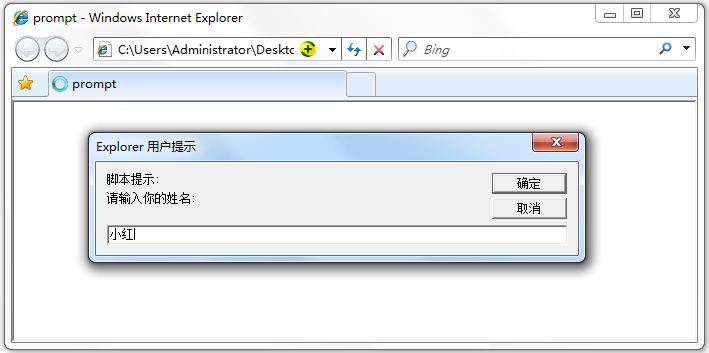
if(myname!=null)

{ alert("你好"+myname); }

else

{ alert("你好 my friend."); }

**结果:**

**[](http://img.mukewang.com/52e360780001ede107090353.jpg)**

**注:在用户点击对话框的按钮前，不能进行任何其它操作。**

2-5 **打开新窗口（window.open）**

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:**

**[](http://img.mukewang.com/52e3677900013d6a05020261.jpg)**

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。

2-6 **关闭窗口（window.close）**

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。

3-1 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:**

**[](http://img.mukewang.com/52e4be610001c67307860420.jpg)**

**将HTML代码分解为DOM节点层次图:**

**[](http://img.mukewang.com/52e4bd0f0001dd8d04830279.jpg)**

**HTML文档可以说由节点构成的集合，三种常见的DOM节点:**

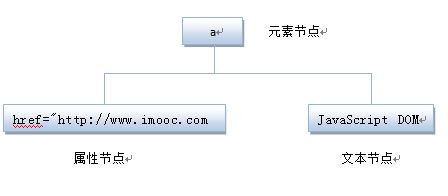
**1. 元素节点：**上图中<html>、<body>、<p>等都是元素节点，即标签。

**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>

[](http://img.mukewang.com/52e4bdb80001064c04480196.jpg)

3-2 **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。

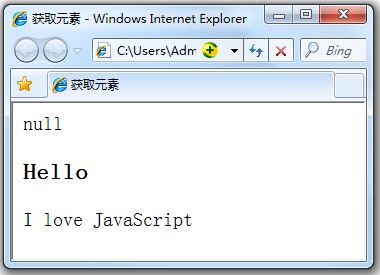
**语法:**

**document.getElementById(“id”)**

**看看下面代码:**

**[](http://img.mukewang.com/52e4c5950001054207900423.jpg)**

**结果:null或[object HTMLParagraphElement]**

**[](http://img.mukewang.com/52e4c6080001734c03800275.jpg)**

**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。**

3-3 **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:**

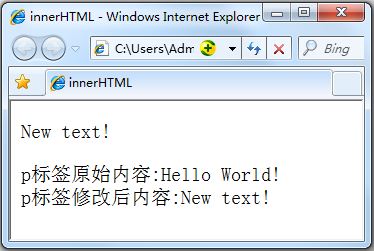
1.Object是获取的元素对象，如通过document.getElementById("ID")获取的元素。

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:**

**[](http://img.mukewang.com/52e4cd080001f01507220418.jpg)**

**结果:**

**[](http://img.mukewang.com/52e4cb5c000187ce03740251.jpg)**

3-4 **改变 HTML 样式**

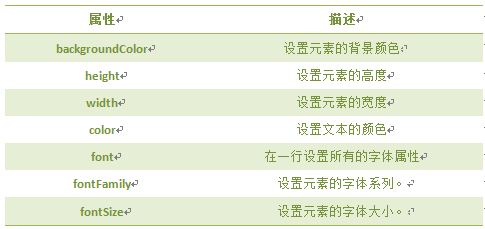
HTML DOM 允许 JavaScript 改变 HTML 元素的样式。如何改变 HTML 元素的样式呢？

**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）:**

**[](http://img.mukewang.com/52e4d4240001dd6c04850229.jpg)**

**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。

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");

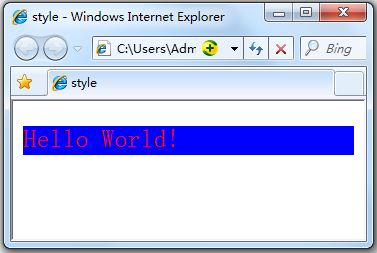
mychar.**style.color**="red";

mychar.**style.fontSize**="20";

mychar.**style.backgroundColor** ="blue";

</script>

**结果:**

[](http://img.mukewang.com/52e4d6ae000177d203770253.jpg)

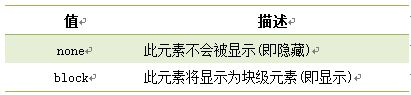
3-5 **显示和隐藏（display属性）**

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:**

**[](http://img.mukewang.com/52e4dba5000179da04110095.jpg)**

**[](http://img.mukewang.com/52e4dcf50001bead09310689.jpg)**

3-6 **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**

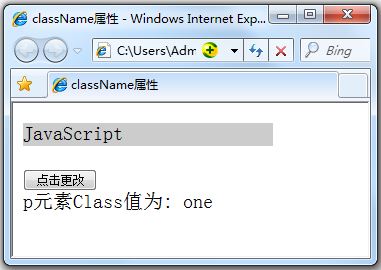
**作用:**

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：**

[](http://img.mukewang.com/52e4e28c0001c97f07980838.jpg)

**结果:[](http://img.mukewang.com/52e4e711000135d903810270.jpg)**

* 1. JS

JavaScript能做什么？

1.增强页面动态效果(如:下拉菜单、图片轮播、信息滚动等)

2.实现页面与用户之间的实时、动态交互(如:用户注册、登陆验证等)

JS进阶篇学习什么？

在JavaScript入门篇中，我们学习了如何插入JS、输出内容及简单的DOM操作，JavaScript进阶篇让您进一步的了解JS的变量、数组、函数、语法、对象、事件、DOM操作，制作简单的网页动态效果。

2-2 变量命名

1.必须以字母、下划线或美元符号开头，后面可以跟字母、下划线、美元符号和数字。如下:

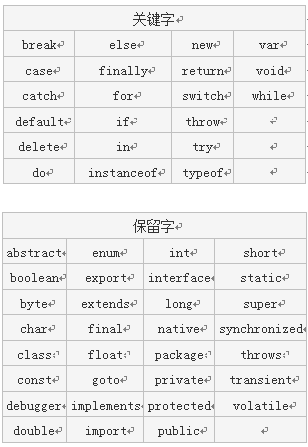
正确: 错误:

mysum 6num //开头不能用数字

\_mychar %sum //开头不能用除(\_ $)外特殊符号,如(% + /等)

$numa1 sum+num //开头中间不能使用除(\_ $)外特殊符号，如(% + /等)

2.变量名区分大小写，如:A与a是两个不同变量。

3.不允许使用JavaScript关键字和保留字做变量名。[](http://img.mukewang.com/529c07c000014f5103080447.jpg)逻辑操作符(&&、||、！)

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

3-1 数组

var myarr=new Array(); //定义数组

myarr[0]=80;

myarr[1]=60; 相当于C++堆中新建数组，根据需要添加更多数值。

**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。

数组赋值：

第一种方法：var myarray = new Array(66,80,90,77,59); //创建数组同时赋值

第二种方法： var myarray = [66,80,90,77,59]; //直接输入一个数组（称 “字面量数组”）

注意：数组存储的数据可以是任何类型（数字、字符、布尔值等）

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

3-6 **数组属性length**

1. myarray.length; //获得数组myarray的长度
2. 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

3-7 **二维数组**

**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

}

}

2. **二维数组的定义方法二**: var Myarr = [[0 , 1 , 2 ],[1 , 2 , 3]]

4-1 判断语句

1. if小写，大写字母（IF）会出错！
2. 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。

4-8 **退出循环break** if(特殊情况)

{break;}

4-9 **继续循环 continue if(特殊情况)**

{ continue; } //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

5-1 **函数** function add2(a,b){

sum = a + b;

alert(sum);

} // 只需写一次就可以

add2(3,2);

add2(7,8); //只需调用函数就可以

1、<input type="button" value="click it" onclick="add2()"> //在HTML文件中通过点击按钮后调用定义好的函数

2、通过“return：返回值”对函数运行结果进行处理

6-1**事件**

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。

**主要事件表:**

[](http://img.mukewang.com/53e198540001b66404860353.jpg)

例：

1、<a href="http://www.imooc.com" onmouseout="message()">点击我</a>

2、光标聚焦事件应用于文本框、下拉列表

3、按钮用表单包括

4、<form>

个人简介：<br>

<textarea name="summary" cols="60" rows="5" onselect="message()">请写入个人简介，不少于200字！</textarea>

</form>

5、加载页面时，触发onload事件，事件写在<body>标签内。

6、onunload事件用于页面关闭、页面刷新等

<script type="text/javascript">

window.onunload = onunload\_message;

function onunload\_message(){

alert("您确定离开该网页吗？");

}

</script>

7-1 **对象**

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

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

7-2 **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：

[](http://img.mukewang.com/555c650d0001ae7b04180297.jpg)

7-3 **返回/设置年份方法**

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。

7-4 **返回星期方法**

**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

7-5 **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);

7-6 **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。

1、访问字符串对象的属性length: stringObject.length; 返回该字符串的长度。

2、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();

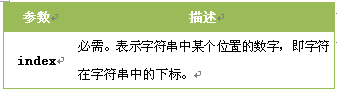
7-7 **返回指定位置的字符**

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：**

[](http://img.mukewang.com/53251a310001cf1e03370092.jpg)  
**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。

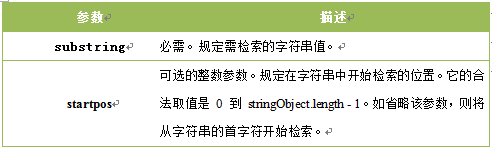
7-8 **返回指定的字符串首次出现的位置**

indexOf() 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：**

[](http://img.mukewang.com/53853d4200019feb04920149.jpg)  
**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

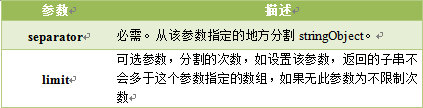
7-9 **字符串分割split()**

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:**

**[](http://img.mukewang.com/532bee4800014c0404230108.jpg)**

**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

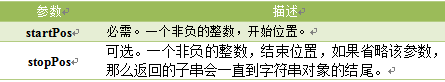
7-10 **提取字符串substring()**

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

**7-11 提取指定数目的字符substr()**

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：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 指倒数第二个字符，以此类推。

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

7-12 **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

[](http://img.mukewang.com/532fe7cf0001e7b505170269.jpg)

Math 对象方法

[](http://img.mukewang.com/532fe841000174db05160622.jpg)

7-13 **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](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。

7-14 **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](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。

7-15 **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：**

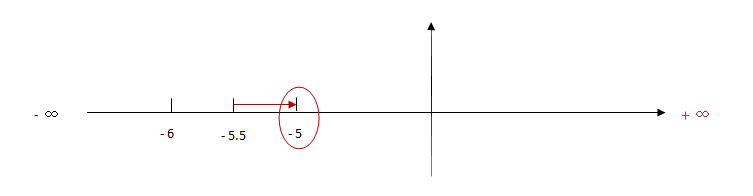
http://img.mukewang.com/532feb70000180ab03210059.jpg

**注意：**

1. 返回与 x 最接近的整数。

2. 对于 0.5，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:

[](http://img.mukewang.com/53fc4cc8000169a907530196.jpg)

7-16 **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();

7-17 **Array 数组对象**

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

**数组定义的方法：**

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>...];

我们定义myArray数组，并赋值，**代码如下：** var myArray = [2, 8, 6];

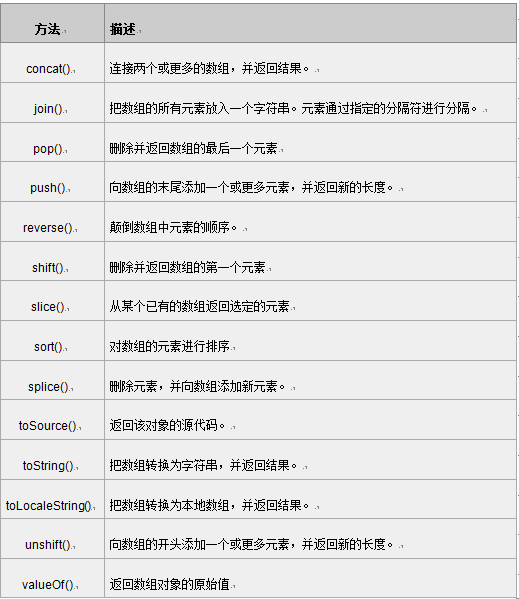
**说明：**定义了一个数组 myArray，里边的元素是：myArray[0] = 2; myArray[1] = 8; myArray[2] = 6。

**数组元素使用：** 数组名[下标] = 值; **注意**: 数组的下标用方括号括起来，从0开始。

**数组属性：**

length 用法：<数组对象>.length；返回：数组的长度，即数组里有多少个元素。它等于数组里最后一个元素的下标加一。

**数组方法：**

[](http://img.mukewang.com/533295ab0001dead05190599.jpg)

7-18 **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,!

7-19 **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)

**参数说明:**

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

7-20 **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] = "1"

myarr[1] = "2"

myarr[2] = "3"

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

document.write(myarr.reverse())

</script>

**运行结果：**

1,2,3

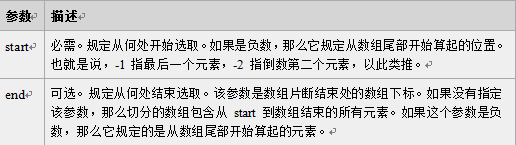
3,2,1

7-21 **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：**

[](http://img.mukewang.com/533299680001637b05160145.jpg)

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);

</script>

**运行结果：**

1,2,3,4,5,6

3,4

1,2,3,4,5,6

7-22 **数组排序sort()**

**sort()**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)

**参数说明：**

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 parseInt() 字符串类型转成整型

8-1 **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:**

**[](http://img.mukewang.com/535483720001a54506670563.jpg)**

**注意:**window对象重点讲解计时器。

8-2 **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：**

[](http://img.mukewang.com/56976e1700014fc504090143.jpg)

8-3 **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>

8-4 **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>

8-5 **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)。

1、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>

2、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>

3、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>

8-6 **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>

8-7 **History 对象**

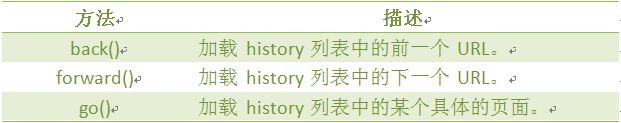
history对象记录了用户曾经浏览过的页面(URL)，并可以实现浏览器前进与后退相似导航的功能。**注意:从窗口被打开的那一刻开始记录，每个浏览器窗口、每个标签页乃至每个框架，都有自己的history对象与特定的window对象关联。**

**语法：**window.history.[属性|方法] **注意：**window可以省略。

**History 对象属性**

[](http://img.mukewang.com/53548c030001759e05840068.jpg)

**History 对象方法**

**[](http://img.mukewang.com/53548c200001228206210123.jpg)**

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>

8-8 **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);

8-9 **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);

8-10 **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：**

[](http://img.mukewang.com/5354947e00011a9a06490153.jpg)

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);

8-11 **Location对象**

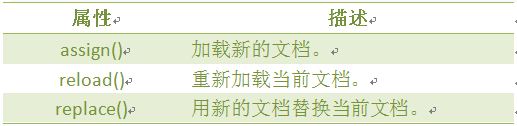
location用于获取或设置窗体的URL，并且可以用于解析URL。

**语法:** location.[属性|方法] 例：location.assign("www.imooc.com") 加载新网页

**location对象属性图示:**

[](http://img.mukewang.com/53605c5a0001b26909900216.jpg)

**location 对象方法:**

**[](http://img.mukewang.com/5354b1eb00016a2405170126.jpg)**

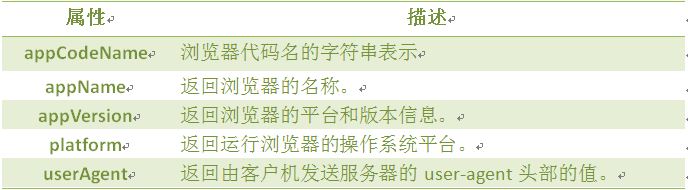
**location 对象属性：**

**[](http://img.mukewang.com/5354b1d00001c4ec06220271.jpg)**

8-12 **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:**

[](http://img.mukewang.com/5354cff70001428b06880190.jpg)

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);

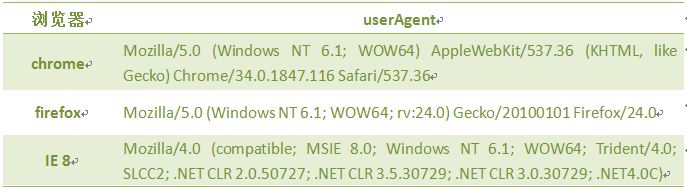
</script>

8-13 **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。

[](http://img.mukewang.com/535a3a4a0001e03f06870189.jpg)

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)";

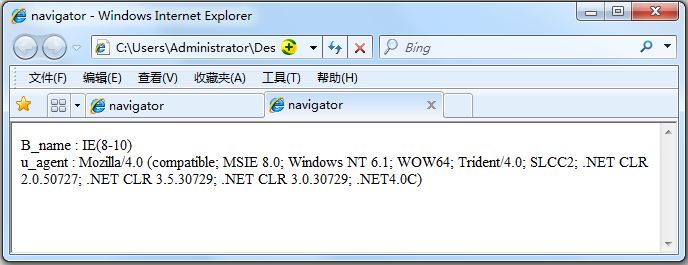
  }

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

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

}

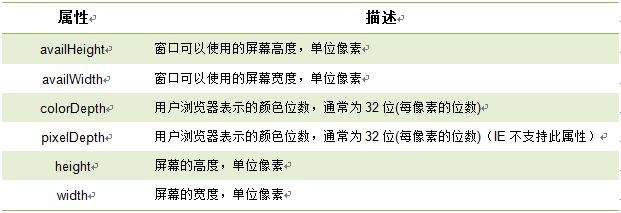
**运行结果:**

[](http://img.mukewang.com/535dea1e00017b0b06880265.jpg)

8-14 **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:**

**[](http://img.mukewang.com/5354d2810001a47706210213.jpg)**

8-15 **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>

8-16 **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);

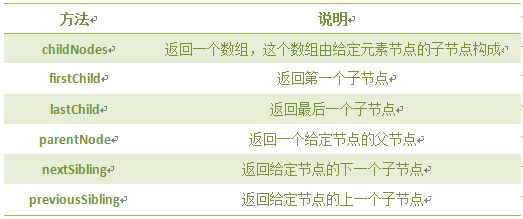
document.write("可用高度：" + screen.availHeight);

</script>

**注意:根据屏幕的不同显示值不同。**

9-1 DOM **节点属性:**

[](http://img.mukewang.com/5375c953000117ee05240129.jpg) **遍历节点树:**

[](http://img.mukewang.com/53f17a6400017d2905230219.jpg)**DOM操作:**

[](http://img.mukewang.com/538d29da000152db05360278.jpg)

**注意:**前两个是document方法。

9-2 **getElementsByName()方法**

返回带有指定名称的节点对象的集合。

**语法：**document.getElementsByName(name)

与getElementById() 方法不同的是，通过元素的 name 属性查询元素，而不是通过 id 属性。

**注意:**

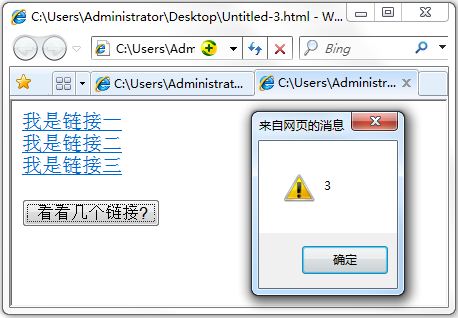
1. 因为文档中的 name 属性可能不唯一，所有 getElementsByName() 方法返回的是元素的数组，而不是一个元素。

2. 和数组类似也有length属性，可以和访问数组一样的方法来访问，从0开始。

**看看下面的代码:**

[](http://img.mukewang.com/5375d5ec00012bac06210322.jpg)

**运行结果:**

[](http://img.mukewang.com/53795f0b0001233404580318.jpg)

9-3 **getElementsByTagName()方法**

返回带有指定标签名的节点对象的集合。返回元素的顺序是它们在文档中的顺序。

**语法:**document.getElementsByTagName(Tagname)

**说明:**

1. Tagname是标签的名称，如p、a、img等标签名。

2. 和数组类似也有length属性，可以和访问数组一样的方法来访问，所以从0开始。

看看下面代码，通过getElementsByTagName()获取节点。

[](http://img.mukewang.com/53ec174a0001404206540436.jpg)

9-4 **区别getElementByID,getElementsByName,getElementsByTagName**

**以人来举例说明，人有能标识身份的身份证，有姓名，有类别(大人、小孩、老人)等。**

1. ID 是一个人的身份证号码，是唯一的。所以通过getElementById获取的是指定的一个人。

2. Name 是他的名字，可以重复。所以通过getElementsByName获取名字相同的人集合。

3. TagName可看似某类，getElementsByTagName获取相同类的人集合。如获取小孩这类人，getElementsByTagName("小孩")。

**把上面的例子转换到HTML中，如下:**

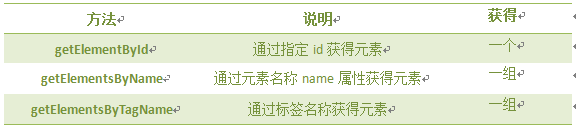
<input type="checkbox" name="hobby" id="hobby1"> 音乐

input标签就像人的类别。

name属性就像人的姓名。

id属性就像人的身份证。

**方法总结如下:**

[](http://img.mukewang.com/5405263300018bcf05760129.jpg)

**注意：**方法区分大小写

通过下面的例子(6个name="hobby"的复选项，两个按钮)来区分三种方法的不同:

<input type="checkbox" name="hobby" id="hobby1"> 音乐

<input type="checkbox" name="hobby" id="hobby2"> 登山

<input type="checkbox" name="hobby" id="hobby3"> 游泳

<input type="checkbox" name="hobby" id="hobby4"> 阅读

<input type="checkbox" name="hobby" id="hobby5"> 打球

<input type="checkbox" name="hobby" id="hobby6"> 跑步

<input type="button" value = "全选" id="button1">

<input type="button" value = "全不选" id="button1">

1. document.getElementsByTagName("input")，结果为获取所有标签为input的元素，共8个。

2. document.getElementsByName("hobby")，结果为获取属性name="hobby"的元素，共6个。

3. document.getElementById("hobby6")，结果为获取属性id="hobby6"的元素，只有一个，"跑步"这个复选项。

9-5 **getAttribute()方法**

通过元素节点的属性名称获取属性的值。

**语法：**elementNode.getAttribute(name)

**说明:**

1. elementNode：使用getElementById()、getElementsByTagName()等方法，获取到的元素节点。

2. name：要想查询的元素节点的属性名字

看看下面的代码，获取h1标签的属性值：

[](http://img.mukewang.com/538d242700019ec809330432.jpg)

**运行结果:**

h1标签的ID ：alink

h1标签的title ：getAttribute()获取标签的属值

9-6 **setAttribute()方法**

setAttribute() 方法增加一个指定名称和值的新属性，或者把一个现有的属性设定为指定的值。

**语法：**elementNode.setAttribute(name,value)

**说明：**1.name: 要设置的属性名。 2.value: 要设置的属性值。

**注意：**1.把指定的属性设置为指定的值。如果不存在具有指定名称的属性，该方法将创建一个新属性。

2.类似于getAttribute()方法，setAttribute()方法只能通过元素节点对象调用的函数。

9-7 **节点属性**

在文档对象模型 (DOM) 中，每个节点都是一个对象。DOM 节点有三个重要的属性 ：

1. nodeName : 节点的名称

2. nodeValue ：节点的值

3. nodeType ：节点的类型

**一、nodeName 属性:**节点的名称，是只读的。

1. 元素节点的 nodeName 与标签名相同  
2. 属性节点的 nodeName 是属性的名称  
3. 文本节点的 nodeName 永远是 #text  
4. 文档节点的 nodeName 永远是 #document

**二、nodeValue 属性：**节点的值

1. 元素节点的 nodeValue 是 undefined 或 null  
2. 文本节点的 nodeValue 是文本自身  
3. 属性节点的 nodeValue 是属性的值

**三、nodeType 属性:**节点的类型，是只读的。以下常用的几种结点类型:

**元素类型    节点类型**  
  元素          1  
  属性          2  
  文本          3  
  注释          8  
  文档          9

9-8 **访问子结点childNodes**

访问选定元素节点下的所有子节点的列表，返回的值可以看作是一个数组，他具有length属性。

**语法：**elementNode.childNodes

**注意：**如果选定的节点没有子节点，则该属性返回不包含节点的 NodeList。

**我们来看看下面的代码:**

[](http://img.mukewang.com/538405fa00010e6c05630357.jpg)

**运行结果:**

**IE:**

UL子节点个数:3

节点类型:1

**其它浏览器:**

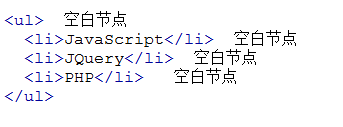
UL子节点个数:7

节点类型:3

**注意:**

1. IE全系列、firefox、chrome、opera、safari兼容问题

2. 节点之间的空白符，在firefox、chrome、opera、safari浏览器是文本节点，所以IE是3，其它浏览器是7，如下图所示:

[](http://img.mukewang.com/538d2b8a000163e303430127.jpg)空白节点没有**nodeValue？**

**如果把代码改成这样:**

<ul><li>javascript</li><li>jQuery</li><li>PHP</li></ul>

**运行结果:（IE和其它浏览器结果是一样的）**

UL子节点个数:3

节点类型:1

9-9 **访问子结点的第一和最后项**

一、firstChild 属性返回‘childNodes’数组的第一个子节点。如果选定的节点没有子节点，则该属性返回 NULL。

**语法：**node.firstChild

**说明：**与elementNode.childNodes[0]是同样的效果。

二、 lastChild 属性返回‘childNodes’数组的最后一个子节点。如果选定的节点没有子节点，则该属性返回 NULL。

**语法：**node.lastChild

**说明：**与elementNode.childNodes[elementNode.childNodes.length-1]是同样的效果。

**注意:**上一节中，我们知道Internet Explorer 会忽略节点之间生成的空白文本节点，而其它浏览器不会。我们可以通过检测节点类型，过滤子节点。 (以后章节讲解)

9-10 **访问父节点parentNode**

获取指定节点的父节点

**语法：**elementNode.parentNode **注意:父节点只能有一个。**

看看下面的例子,获取 P 节点的父节点，代码如下:

<div id="text">

<p id="con"> parentNode 获取指点节点的父节点</p>

</div>

<script type="text/javascript">

  var mynode= document.getElementById("con");

  document.write(mynode.**parentNode**.nodeName);

</script>

运行结果:

parentNode 获取指点节点的父节点

DIV

**访问祖节点:**elementNode.parentNode.parentNode

看看下面的代码:

<div id="text">

  <p>

    parentNode

    <span id="con"> 获取指点节点的父节点</span>

  </p>

</div>

<script type="text/javascript">

  var mynode= document.getElementById("con");

  document.write(mynode.**parentNode.parentNode.**nodeName);

</script>

运行结果:

parentNode获取指点节点的父节点

DIV

注意: 浏览器兼容问题，chrome、firefox等浏览器标签之间的空白也算是一个文本节点。

9-11 **访问兄弟节点**

1. nextSibling 属性可返回某个节点之后紧跟的节点（处于同一树层级中）。

**语法：**nodeObject.nextSibling **说明：**如果无此节点，则该属性返回 null。

2. previousSibling 属性可返回某个节点之前紧跟的节点（处于同一树层级中）。

**语法：**nodeObject.previousSibling **说明：**如果无此节点，则该属性返回 null。

注意: 两个属性获取的是节点。Internet Explorer 会忽略节点间生成的空白文本节点（例如，换行符号），而其它浏览器不会忽略。

**解决问题方法:**判断节点nodeType是否为1, 如是为元素节点，跳过。

[](http://img.mukewang.com/5386e3c80001c25607010856.jpg)

**运行结果:**

LI = javascript

nextsibling: LI = jquery

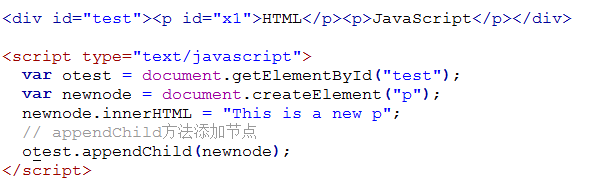
9-12 **插入节点appendChild()**

在指定节点的最后一个子节点列表之后添加一个新的子节点。

**语法:**appendChild(newnode)

**参数:** newnode：指定追加的节点。

我们来看看，div标签内创建一个新的 P 标签，代码如下:

[](http://img.mukewang.com/5398fd020001ad4905890193.jpg)

**运行结果:**

HTML

JavaScript

This is a new p

9-13 **插入节点insertBefore()**

insertBefore() 方法可在已有的子节点前插入一个新的子节点。

**语法:**insertBefore(newnode,node);

**参数:**

newnode: 要插入的新节点。

node: 指定此节点前插入节点。

我们在来看看下面代码，在指定节点前插入节点。

[](http://img.mukewang.com/5395318100010c6806960431.jpg)

**运行结果:**

This is a new p

JavaScript

HTML

注意: otest.insertBefore(newnode,node); 也可以改为:otest.insertBefore(newnode,otest.childNodes[0]);

9-14 **删除节点removeChild()**

removeChild() 方法从子节点列表中删除某个节点。如删除成功，此方法可返回被删除的节点，如失败，则返回 NULL。

**语法:**nodeObject.removeChild(node)

**参数:**node ：必需，指定需要删除的节点。

我们来看看下面代码，删除子点。

[](http://img.mukewang.com/5399744d000153a306060342.jpg)

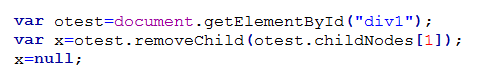
**运行结果:**

HTML

删除节点的内容: javascript

**注意:** 把删除的子节点赋值给 x，这个子节点不在DOM树中，但是还存在内存中，可通过 x 操作。

如果要完全删除对象，给 x 赋 null 值，代码如下:

[](http://img.mukewang.com/539975a800017c8e04790082.jpg)

9-15 **替换元素节点replaceChild()**

replaceChild 实现子节点(对象)的替换。返回被替换对象的引用。

**语法：**node.replaceChild (newnode,oldnew )

**参数：**

newnode : 必需，用于替换 oldnew 的对象。   
oldnew : 必需，被 newnode 替换的对象。

我们来看看下面的代码:

[](http://img.mukewang.com/539557d70001c3ee07190429.jpg) 效果: 将文档中的 Java 改为 JavaScript。

**注意:**

1. 当 oldnode 被替换时，所有与之相关的属性内容都将被移除。

2. newnode 必须先被建立。

9-16 **创建元素节点createElement**

createElement()方法可创建元素节点。此方法可返回一个 Element 对象。

**语法：**document.createElement(tagName)

**参数:**tagName：字符串值，这个字符串用来指明创建元素的类型。

**注意：**要与appendChild() 或 insertBefore()方法联合使用，将元素显示在页面中。

我们来创建一个按钮，代码如下：

<script type="text/javascript">

var body = document.body;

var input = document.createElement("input");

input.type = "button";

input.value = "创建一个按钮";

body.appendChild(input);

</script>

效果：在HTML文档中，创建一个按钮。

我们也可以使用setAttribute来设置属性，代码如下：

<script type="text/javascript">

var body= document.body;

var btn = document.createElement("input");

btn.setAttribute("type", "text");

btn.setAttribute("name", "q");

btn.setAttribute("value", "使用setAttribute");

btn.setAttribute("onclick", "javascript:alert('This is a text!');");

body.appendChild(btn);

</script>

效果：在HTML文档中，创建一个文本框，使用setAttribute设置属性值。 当点击这个文本框时，会弹出对话框“This is a text!”。

9-17 **创建文本节点createTextNode**

createTextNode() 方法创建新的文本节点，返回新创建的 Text 节点。

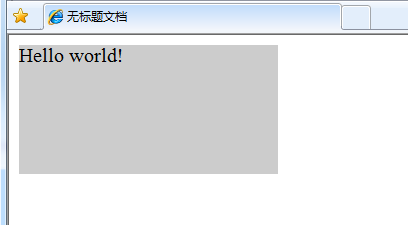
**语法：**document.createTextNode(data)

**参数：**data : 字符串值，可规定此节点的文本。

我们来创建一个<div>元素并向其中添加一条消息，代码如下：

[](http://img.mukewang.com/53951c200001d32d07130554.jpg)

**运行结果:**

[](http://img.mukewang.com/53951c720001996d04080225.jpg)

9-18 **浏览器窗口可视区域大小**

获得浏览器窗口的尺寸（浏览器的视口，不包括工具栏和滚动条）的方法:

**一、对于IE9+、Chrome、Firefox、Opera 以及 Safari：**

window.innerHeight - 浏览器窗口的内部高度

window.innerWidth - 浏览器窗口的内部宽度

**二、对于 Internet Explorer 8、7、6、5：**

document.documentElement.clientHeight表示HTML文档所在窗口的当前高度。

 document.documentElement.clientWidth表示HTML文档所在窗口的当前宽度。

或者

Document对象的body属性对应HTML文档的<body>标签

document.body.clientHeight

document.body.clientWidth

**在不同浏览器都实用的 JavaScript 方案：**

var w= document.documentElement.clientWidth

|| document.body.clientWidth;

var h= document.documentElement.clientHeight

|| document.body.clientHeight; 注：360浏览器用前者

9-19 **网页尺寸scrollHeight**

scrollHeight和scrollWidth，获取网页内容高度和宽度。

**一、针对IE、Opera:**

scrollHeight 是网页内容实际高度，可以小于 clientHeight。

**二、针对NS、FF:**

scrollHeight 是网页内容高度，不过最小值是 clientHeight。也就是说网页内容实际高度小于 clientHeight 时，scrollHeight 返回 clientHeight 。

**三、浏览器兼容性**

var w=document.documentElement.scrollWidth|| document.body.scrollWidth;

var h=document.documentElement.scrollHeight|| document.body.scrollHeight;

注：360浏览器用后者

**注意:区分大小写**

scrollHeight和scrollWidth还可获取Dom元素中内容实际占用的高度和宽度。

9-20 **网页尺寸offsetHeight**

offsetHeight和offsetWidth，获取网页内容高度和宽度(包括滚动条等边线，会随窗口的显示大小改变)。

**一、值**

offsetHeight = clientHeight + 滚动条 + 边框。

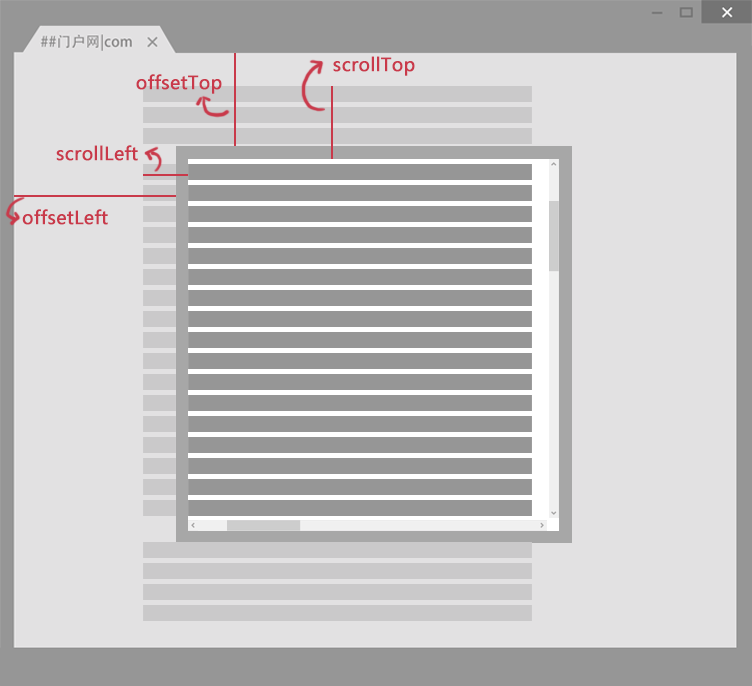
**二、浏览器兼容性**

var w= document.documentElement.offsetWidth || document.body.offsetWidth;

var h= document.documentElement.offsetHeight || document.body.offsetHeight;

9-21 **网页卷去的距离与偏移量**

**我们先来看看下面的图：**

[](http://img.mukewang.com/5347b2b10001e1a307520686.jpg)

**scrollLeft:**设置或获取位于给定对象左边界与窗口中目前可见内容的最左端之间的距离 ，即左边灰色的内容。

**scrollTop:**设置或获取位于对象最顶端与窗口中可见内容的最顶端之间的距离 ，即上边灰色的内容。

**offsetLeft:**获取指定对象相对于版面或由 offsetParent 属性指定的父坐标的计算左侧位置 。

**offsetTop:**获取指定对象相对于版面或由 offsetParent 属性指定的父坐标的计算顶端位置 。

**注意:**

**1. 区分大小写**

**2. offsetParent：布局中设置postion属性(Relative、Absolute、fixed)的父容器，从最近的父节点开始，一层层向上找，直到HTML的body。**