Java script学习：

一、JavaScript 基础

### 1.1 JavaScript简介

**这门语言可用于 HTML 和 web，更可广泛用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。**

#### **JavaScript 是脚本语言**

JavaScript 是一种轻量级的编程语言。

JavaScript 是可插入 HTML 页面的编程代码。

JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。

### 1.2 JavaScript基础

**HTML 中的脚本必须位于 <script> 与 </script> 标签之间。**

**脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。**

#### 如需在 HTML 页面中插入 JavaScript，请使用 <script> 标签。

<script> 和 </script> 会告诉 JavaScript 在何处开始和结束。

<script> 和 </script> 之间的代码行包含了 JavaScript：

<script>

alert("My First JavaScript");

</script>

那些老旧的实例可能会在 <script> 标签中使用 type="text/javascript"。现在已经不必这样做了。JavaScript 是所有现代浏览器以及 HTML5 中的默认脚本语言。

#### **外部的 JavaScript**

也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件：

<!DOCTYPE html>

<html>

<body>

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

</body>

</html>

### 1.3 JavaScript 输出

#### **JavaScript 通常用于操作 HTML 元素。**

#### **操作 HTML 元素**

如需从 JavaScript 访问某个 HTML 元素，您可以使用 document.getElementById(*id*) 方法。

请使用 "id" 属性来标识 HTML 元素：

通过指定的 id 来访问 HTML 元素，并改变其内容：

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p id="demo">My First Paragraph</p>

<script>

document.getElementById("demo").innerHTML="My First JavaScript";

</script>

</body>

</html>

#### **写到文档输出**

下面的例子直接把 <p> 元素写到 HTML 文档输出中：

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<script>

document.write("<p>My First JavaScript</p>");

</script>

</body>

</html>

****警告****

请使用 document.write() 仅仅向文档输出写内容。

如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My First Paragraph.</p>

<button onclick="myFunction()">点击这里</button>

<script>

function myFunction()

{

document.write("糟糕！文档消失了。");

}

</script>

</body>

</html>

#### **Windows 8 中的 JavaScript**

**提示：**微软支持通过 JavaScript 创建 Windows 8 app。

对于因特网和视窗操作系统，JavaScript 都意味着未来。

### 1.4 JavaScript 语句

#### **JavaScript 语句**

JavaScript 语句向浏览器发出的命令。语句的作用是告诉浏览器该做什么。

#### **分号 ;**

分号用于分隔 JavaScript 语句。

通常我们在每条可执行的语句结尾添加分号。

使用分号的另一用处是在一行中编写多条语句。

**提示：**您也可能看到不带有分号的案例。

在 JavaScript 中，用分号来结束语句是可选的。

#### **JavaScript 代码**

JavaScript 代码（或者只有 JavaScript）是 JavaScript 语句的序列。

浏览器会按照编写顺序来执行每条语句。

#### **JavaScript 代码块**

JavaScript 语句通过代码块的形式进行组合。

块由左花括号开始，由右花括号结束。

块的作用是使语句序列一起执行。

JavaScript 函数是将语句组合在块中的典型例子。

function myFunction()

{

document.getElementById("demo").innerHTML="Hello World";

document.getElementById("myDIV").innerHTML="How are you?";

}

#### **JavaScript 对大小写敏感。**

JavaScript 对大小写是敏感的。

当编写 JavaScript 语句时，请留意是否关闭大小写切换键。

函数 getElementById 与 getElementbyID 是不同的。

同样，变量 myVariable 与 MyVariable 也是不同的。

#### **空格**

JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。

#### **对代码行进行折行**

您可以在文本字符串中使用反斜杠对代码行进行换行。下面的例子会正确地显示：

document.write("Hello \

World!");

不过，您不能像这样折行：

document.write \

("Hello World!");

**提示：**JavaScript 是脚本语言。浏览器会在读取代码时，逐行地执行脚本代码。而对于传统编程来说，会在执行前对所有代码进行编译。

### 1.5 JavaScript 注释

**JavaScript 注释可用于提高代码的可读性。**

单行注释以 // 开头。

多行注释以 /\* 开始，以 \*/ 结尾。

### 1.6JavaScript 变量

**变量是存储信息的容器。**

var x=2;

var y=3;

var z=x+y;

 变量必须以字母开头

 变量也能以 $ 和 \_ 符号开头（不过我们不推荐这么做）

 变量名称对大小写敏感（y 和 Y 是不同的变量）

#### **JavaScript 数据类型**

JavaScript 变量还能保存其他数据类型，比如文本值 (name="Bill Gates")。

在 JavaScript 中，类似 "Bill Gates" 这样一条文本被称为字符串。

JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

var pi=3.14;

var name="Bill Gates";

var answer='Yes I am!';

#### **声明（创建） JavaScript 变量**

在 JavaScript 中创建变量通常称为“声明”变量。

我们使用 var 关键词来声明变量：

var carname;

变量声明之后，该变量是空的（它没有值）。

如需向变量赋值，请使用等号：

carname="Volvo";

不过，您也可以在声明变量时对其赋值：

var carname="Volvo";

**提示：**一个好的编程习惯是，在代码开始处，统一对需要的变量进行声明。

#### **一条语句，多个变量**

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

var name="Gates", age=56, job="CEO";

声明也可横跨多行：

var name="Gates",

age=56,

job="CEO";

#### **Value = undefined**

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。

在执行过以下语句后，变量 carname 的值将是 undefined：

var carname;

#### **重新声明 JavaScript 变量**

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

var carname="Volvo";

var carname;

#### **JavaScript 算数**

您可以通过 JavaScript 变量来做算数，使用的是 = 和 + 这类运算符：

y=5;

x=y+2;

### 1.7JavaScript 数据类型

#### 字符串、数字、布尔、数组、对象、Null、Undefined

#### **JavaScript 拥有动态类型**

JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

var x // x 为 undefined

var x = 6; // x 为数字

var x = "Bill"; // x 为字符串

#### **JavaScript 字符串**

字符串是存储字符（比如 "Bill Gates"）的变量。

字符串可以是引号中的任意文本。您可以使用单引号或双引号：

var carname="Bill Gates";

var carname='Bill Gates';

您可以在字符串中使用引号，只要不匹配包围字符串的引号即可：

var answer="Nice to meet you!";

var answer="He is called 'Bill'";

var answer='He is called "Bill"';

#### **JavaScript 数字**

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带

var x1=34.00; //使用小数点来写

var x2=34; //不使用小数点来写

极大或极小的数字可以通过科学（指数）计数法来书写：

var y=123e5; // 12300000

var z=123e-5; // 0.00123

#### **JavaScript 布尔**

布尔（逻辑）只能有两个值：true 或 false。

var x=true

var y=false

#### **JavaScript 数组**

下面的代码创建名为 cars 的数组：

var cars=new Array();

cars[0]="Audi";

cars[1]="BMW";

cars[2]="Volvo";

或者 (condensed array):

var cars=new Array("Audi","BMW","Volvo");

或者 (literal array):

var cars=["Audi","BMW","Volvo"];

数组下标是基于零的，所以第一个项目是 [0]，第二个是 [1]，以此类推。

#### **JavaScript 对象**

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

var person={firstname:"Bill", lastname:"Gates", id:5566};

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。

空格和折行无关紧要。声明可横跨多行：

var person={

firstname : "Bill",

lastname : "Gates",

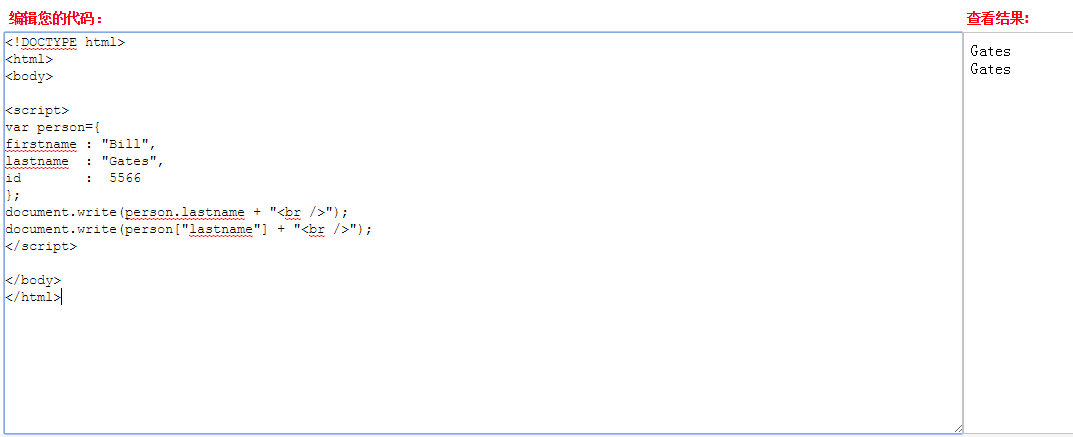
id : 5566

};

对象属性有两种寻址方式：

name=person.lastname;

name=person["lastname"];



#### **声明变量类型**

当您声明新变量时，可以使用关键词 "new" 来声明其类型：

var carname=new String;

var x= new Number;

var y= new Boolean;

var cars= new Array;

var person= new Object;

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

### 1.8 JavaScript 对象

**JavaScript 中的所有事物都是对象：字符串、数字、数组、日期，等等。**

**在 JavaScript 中，对象是拥有属性和方法的数据。**

#### 属性和方法

属性是与对象相关的值。

方法是能够在对象上执行的动作。

#### JavaScript 中的对象

在 JavaScript 中，对象是数据（变量），拥有属性和方法。

当您像这样声明一个 JavaScript 变量时：

var txt = "Hello";

您实际上已经创建了一个 JavaScript 字符串对象。字符串对象拥有内建的属性 length。对于上面的字符串来说，length 的值是 5。字符串对象同时拥有若干个内建的方法。

属性：

txt.length=5

方法：

txt.indexOf()

txt.replace()

txt.search()

**提示：**在面向对象的语言中，属性和方法常被称为对象的成员。

#### 创建 JavaScript 对象

JavaScript 中的几乎所有事务都是对象：字符串、数字、数组、日期、函数，等等。

你也可以创建自己的对象。

本例创建名为 "person" 的对象，并为其添加了四个属性：

person=new Object();

person.firstname="Bill";

person.lastname="Gates";

person.age=56;

person.eyecolor="blue";

#### 访问对象的属性

访问对象属性的语法是：

*objectName*.*propertyName*

#### 访问对象的方法

您可以通过下面的语法调用方法：

*objectName*.*methodName*()

**提示：**在面向对象的语言中，使用 camel-case 标记法的函数是很常见的。您会经常看到 someMethod() 这样的函数名，而不是 some\_method()。

### 1.9 JavaScript 函数

**函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。**

#### JavaScript 函数语法

函数就是包裹在花括号中的代码块，前面使用了关键词 function：

function functionname()

{

这里是要执行的代码

}

当调用该函数时，会执行函数内的代码。

可以在某事件发生时直接调用函数（比如当用户点击按钮时），并且可由 JavaScript 在任何位置进行调用。

**提示：**JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

#### 调用带参数的函数

在调用函数时，您可以向其传递值，这些值被称为参数。

这些参数可以在函数中使用。

您可以发送任意多的参数，由逗号 (,) 分隔：

myFunction(*argument1*,*argument2*)

当您声明函数时，请把参数作为变量来声明：

function myFunction(var1,var2)

{

这里是要执行的代码

}

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递的参数的给定的值，以此类推。

<button onclick="myFunction('Bill Gates','CEO')">点击这里</button>

<script>

function myFunction(name,job)

{

alert("Welcome " + name + ", the " + job);

}

</script>

#### 带有返回值的函数

有时，我们会希望函数将值返回调用它的地方。

通过使用 return 语句就可以实现。

在使用 return 语句时，函数会停止执行，并返回指定的值。

#### 语法

function myFunction()

{

var x=5;

return x;

}

上面的函数会返回值 5。

**注释：**整个 JavaScript 并不会停止执行，仅仅是函数。JavaScript 将继续执行代码，从调用函数的地方。

函数调用将被返回值取代：

var myVar=myFunction();

myVar 变量的值是 5，也就是函数 "myFunction()" 所返回的值。

即使不把它保存为变量，您也可以使用返回值：

document.getElementById("demo").innerHTML=myFunction();

"demo" 元素的 innerHTML 将成为 5，也就是函数 "myFunction()" 所返回的值。

您可以使返回值基于传递到函数中的参数：

在您仅仅希望退出函数时 ，也可使用 return 语句。返回值是可选的：

#### 局部 JavaScript 变量

在 JavaScript 函数内部声明的变量（使用 var）是**局部**变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。

您可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。

只要函数运行完毕，本地变量就会被删除。

#### 全局 JavaScript 变量

在函数外声明的变量是**全局**变量，网页上的所有脚本和函数都能访问它。

#### JavaScript 变量的生存期

JavaScript 变量的生命期从它们被声明的时间开始。

局部变量会在函数运行以后被删除。

全局变量会在页面关闭后被删除。

#### 向未声明的 JavaScript 变量来分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

这条语句：

carname="Volvo";

将声明一个**全局**变量 carname，即使它在函数内执行。

### 1.10 JavaScript 运算符

#### JavaScript 算术运算符

算术运算符用于执行变量与/或值之间的算术运算。

给定 **y=5**，下面的表格解释了这些算术运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **描述** | **例子** | **结果** |
| + | 加 | x=y+2 | x=7 |
| - | 减 | x=y-2 | x=3 |
| \* | 乘 | x=y\*2 | x=10 |
| / | 除 | x=y/2 | x=2.5 |
| % | 求余数 (保留整数) | x=y%2 | x=1 |
| ++ | 累加 | x=++y | x=6 |
| -- | 递减 | x=--y | x=4 |

#### JavaScript 赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

给定 **x=10** 和 **y=5**，下面的表格解释了赋值运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **例子** | **等价于** | **结果** |
| = | x=y |  | x=5 |
| += | x+=y | x=x+y | x=15 |
| -= | x-=y | x=x-y | x=5 |
| \*= | x\*=y | x=x\*y | x=50 |
| /= | x/=y | x=x/y | x=2 |
| %= | x%=y | x=x%y | x=0 |

#### 用于字符串的 + 运算符

+ 运算符用于把文本值或字符串变量加起来（连接起来）。

如需把两个或多个字符串变量连接起来，请使用 + 运算符。

#### 对字符串和数字进行加法运算

**规则是：**

**如果把数字与字符串相加，结果将成为字符串。**

#### 比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

给定 x=5，下面的表格解释了比较运算符：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| == | 等于 | x==8 为 false |
| === | 全等（值和类型） | x===5 为 true；x==="5" 为 false |
| != | 不等于 | x!=8 为 true |
| > | 大于 | x>8 为 false |
| < | 小于 | x<8 为 true |
| >= | 大于或等于 | x>=8 为 false |
| <= | 小于或等于 | x<=8 为 true |

#### 逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| && | and | (x < 10 && y > 1) 为 true |
| || | or | (x==5 || y==5) 为 false |
| ! | not | !(x==y) 为 true |

#### 条件运算符

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

**语法**

variablename=(condition)?value1:value2

**例子**

greeting=(visitor=="PRES")?"Dear President ":"Dear ";

如果变量 visitor 中的值是 "PRES"，则向变量 greeting 赋值 "Dear President "，否则赋值 "Dear"。

### 1.11 JavaScript 条件语句（If...Else、switch ）

**条件语句用于基于不同的条件来执行不同的动作。**

#### 条件语句

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中，我们可使用以下条件语句：

* **if 语句** - 只有当指定条件为 true 时，使用该语句来执行代码
* **if...else 语句** - 当条件为 true 时执行代码，当条件为 false 时执行其他代码
* **if...else if....else 语句** - 使用该语句来选择多个代码块之一来执行
* **switch 语句** - 使用该语句来选择多个代码块之一来执行

#### If 语句

只有当指定条件为 true 时，该语句才会执行代码。

**语法**

if (条件)

{

只有当条件为 true 时执行的代码

}

**注意：**请使用小写的 if。使用大写字母（IF）会生成 JavaScript 错误！

#### If...else 语句

请使用 if....else 语句在条件为 true 时执行代码，在条件为 false 时执行其他代码。

**语法**

if (条件)

{

当条件为 true 时执行的代码

}

else

{

当条件不为 true 时执行的代码

}

#### If...else if...else 语句

使用 if....else if...else 语句来选择多个代码块之一来执行。

**语法**

if (条件 1)

{

当条件 1 为 true 时执行的代码

}

else if (条件 2)

{

当条件 2 为 true 时执行的代码

}

else

{

当条件 1 和 条件 2 都不为 true 时执行的代码

}

#### JavaScript Switch 语句

请使用 switch 语句来选择要执行的多个代码块之一。

**语法**

switch(n)

{

case 1:

执行代码块 1

break;

case 2:

执行代码块 2

break;

default:

n 与 case 1 和 case 2 不同时执行的代码

}

工作原理：首先设置表达式 n（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 **break** 来阻止代码自动地向下一个 case 运行。

#### default 关键词

请使用 default 关键词来规定匹配不存在时做的事情

### 1.12 JavaScript 循环

#### JavaScript 循环

如果您希望一遍又一遍地运行相同的代码，并且每次的值都不同，那么使用循环是很方便的。

#### 不同类型的循环

JavaScript 支持不同类型的循环：

* **for** - 循环代码块一定的次数
* **for/in** - 循环遍历对象的属性
* **while** - 当指定的条件为 true 时循环指定的代码块
* **do/while** - 同样当指定的条件为 true 时循环指定的代码块

#### For 循环

for 循环是您在希望创建循环时常会用到的工具。

下面是 for 循环的语法：

for (语句 1; 语句 2; 语句 3)

{

被执行的代码块

}

**语句 1** 在循环（代码块）开始前执行

**语句 2** 定义运行循环（代码块）的条件

**语句 3** 在循环（代码块）已被执行之后执行

#### For/In 循环

JavaScript for/in 语句循环遍历对象的属性：

**实例**

var person={fname:"John",lname:"Doe",age:25};

for (x in person)

{

txt=txt + person[x];

}

#### while 循环

While 循环会在指定条件为真时循环执行代码块。

**语法**

while (条件)

{

需要执行的代码

}

**提示：**如果您忘记增加条件中所用变量的值，该循环永远不会结束。该可能导致浏览器崩溃。

#### do/while 循环

do/while 循环是 while 循环的变体。该循环会执行一次代码块，在检查条件是否为真之前，然后如果条件为真的话，就会重复这个循环。

**语法**

do

{

需要执行的代码

}

while (条件);

#### 比较 for 和 while

如果您已经阅读了前面那一章关于 for 循环的内容，您会发现 while 循环与 for 循环很像。

**for 语句实例**

本例中的循环使用 for 循环来显示 cars 数组中的所有值：

cars=["BMW","Volvo","Saab","Ford"];

var i=0;

for (;cars[i];)

{

document.write(cars[i] + "<br>");

i++;

}

**while 语句实例**

本例中的循环使用使用 while 循环来显示 cars 数组中的所有值：

cars=["BMW","Volvo","Saab","Ford"];

var i=0;

while (cars[i])

{

document.write(cars[i] + "<br>");

i++;

}

### 1.13 JavaScript Break 和 Continue 语句

**break 语句用于跳出循环。**

**continue 用于跳过循环中的一个迭代。**

**JavaScript 标签**

如需标记 JavaScript 语句，请在语句之前加上冒号：

label:

语句

break 和 continue 语句仅仅是能够跳出代码块的语句。

**语法**

break labelname;

continue labelname;

continue 语句（带有或不带标签引用）只能用在循环中。

break 语句（不带标签引用），只能用在循环或 switch 中。

通过标签引用，break 语句可用于跳出任何 JavaScript 代码块

#### 1.14 JavaScript 错误 - Throw、Try 和 Catch

**try** 语句测试代码块的错误。

**catch** 语句处理错误。

**throw** 语句创建自定义错误。

#### 错误一定会发生

当 JavaScript 引擎执行 JavaScript 代码时，会发生各种错误：

可能是语法错误，通常是程序员造成的编码错误或错别字。

可能是拼写错误或语言中缺少的功能（可能由于浏览器差异）。

可能是由于来自服务器或用户的错误输出而导致的错误。

当然，也可能是由于许多其他不可预知的因素。

#### JavaScript 抛出错误

当错误发生时，当事情出问题时，JavaScript 引擎通常会停止，并生成一个错误消息。

描述这种情况的技术术语是：JavaScript 将**抛出**一个错误。

#### JavaScript 测试和捕捉

**try** 语句允许我们定义在执行时进行错误测试的代码块。

**catch** 语句允许我们定义当 try 代码块发生错误时，所执行的代码块。

JavaScript 语句 **try** 和 **catch** 是成对出现的。

**语法**

try

{

//在这里运行代码

}

catch(err)

{

//在这里处理错误

}

在下面的例子中，我们故意在 try 块的代码中写了一个错字。

catch 块会捕捉到 try 块中的错误，并执行代码来处理它。

<!DOCTYPE html>

<html>

<head>

<script>

var txt="";

function message()

{

try

{

adddlert("Welcome guest!");

}

catch(err)

{

txt="There was an error on this page.\n\n";

txt+="Error description: " + err.message + "\n\n";

txt+="Click OK to continue.\n\n";

alert(txt);

}

}

</script>

</head>

<body>

<input type="button" value="View message" onclick="message()">

</body>

</html>

#### Throw 语句

throw 语句允许我们创建自定义错误。

正确的技术术语是：创建或**抛出异常**（exception）。

如果把 throw 与 try 和 catch 一起使用，那么您能够控制程序流，并生成自定义的错误消息。

**语法**

throw *exception*

异常可以是 JavaScript 字符串、数字、逻辑值或对象。

本例检测输入变量的值。如果值是错误的，会抛出一个异常（错误）。catch 会捕捉到这个错误，并显示一段自定义的错误消息：

<script>

function myFunction()

{

try

{

var x=document.getElementById("demo").value;

if(x=="") throw "empty";

if(isNaN(x)) throw "not a number";

if(x>10) throw "too high";

if(x<5) throw "too low";

}

catch(err)

{

var y=document.getElementById("mess");

y.innerHTML="Error: " + err + ".";

}

}

</script>

<h1>My First JavaScript</h1>

<p>Please input a number between 5 and 10:</p>

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

<button type="button" onclick="myFunction()">Test Input</button>

<p id="mess"></p>

请注意，如果 getElementById 函数出错，上面的例子也会抛出一个错误。

### 1.15 JavaScript 表单验证

**JavaScript 可用来在数据被送往服务器前对 HTML 表单中的这些输入数据进行验证。**

#### JavaScript 表单验证

JavaScript 可用来在数据被送往服务器前对 HTML 表单中的这些输入数据进行验证。

被 JavaScript 验证的这些典型的表单数据有：

* 用户是否已填写表单中的必填项目？
* 用户输入的邮件地址是否合法？
* 用户是否已输入合法的日期？
* 用户是否在数据域 (numeric field) 中输入了文本？

#### 必填（或必选）项目

下面的函数用来检查用户是否已填写表单中的必填（或必选）项目。假如必填或必选项为空，那么警告框会弹出，并且函数的返回值为 false，否则函数的返回值则为 true（意味着数据没有问题）：

function validate\_required(field,alerttxt)

{

with (field)

{

if (value==null||value=="")

{alert(alerttxt);return false}

else {return true}

}

}

下面是连同 HTML 表单的代码：

<html>

<head>

<script type="text/javascript">

function validate\_required(field,alerttxt)

{

with (field)

{

if (value==null||value=="")

{alert(alerttxt);return false}

else {return true}

}

}

function validate\_form(thisform)

{

with (thisform)

{

if (validate\_required(email,"Email must be filled out!")==false)

{email.focus();return false}

}

}

</script>

</head>

<body>

<form action="submitpage.htm" onsubmit="return validate\_form(this)" method="post">

Email: <input type="text" name="email" size="30">

<input type="submit" value="Submit">

</form>

</body>

</html>

#### E-mail 验证

下面的函数检查输入的数据是否符合电子邮件地址的基本语法。

意思就是说，输入的数据必须包含 @ 符号和点号(.)。同时，@ 不可以是邮件地址的首字符，并且 @ 之后需有至少一个点号：

function validate\_email(field,alerttxt)

{

with (field)

{

apos=value.indexOf("@")

dotpos=value.lastIndexOf(".")

if (apos<1||dotpos-apos<2)

{alert(alerttxt);return false}

else {return true}

}

}

下面是连同 HTML 表单的完整代码：

<html>

<head>

<script type="text/javascript">

function validate\_email(field,alerttxt)

{

with (field)

{

apos=value.indexOf("@")

dotpos=value.lastIndexOf(".")

if (apos<1||dotpos-apos<2)

{alert(alerttxt);return false}

else {return true}

}

}

function validate\_form(thisform)

{

with (thisform)

{

if (validate\_email(email,"Not a valid e-mail address!")==false)

{email.focus();return false}

}

}

</script>

</head>

<body>

<form action="submitpage.htm"onsubmit="return validate\_form(this);" method="post">

Email: <input type="text" name="email" size="30">

<input type="submit" value="Submit">

</form>

</body>

</html>

#### js的with语句使用方法

1）简要说明    
       with 语句可以方便地用来引用某个特定对象中已有的属性，但是不能用来给对象添加属性。要给对象创建新的属性，必须明确地引用该对象。    
  
2）语法格式

 with(object instance)

{

//代码块

}

       有时候，我在一个程序代码中，多次需要使用某对象的属性或方法，照以前的写法，都是通过:对象.属性或者对象.方法这样的方式来分别获得该对象的属性和方法，着实有点麻烦，学习了with语句后，可以通过类似如下的方式来实现：  

with(objInstance)    
{    
       var str = 属性1;    
.....    
}

 去除了多次写对象名的麻烦。    
  
3）举例

<script language="javascript">    
<!--    
function Lakers() {    
       this.name = "kobe bryant";    
       this.age = "28";    
       this.gender = "boy";    
}    
var people=new Lakers();    
with(people)    
{    
       var str = "姓名: " + name + "<br>";    
       str += "年龄：" + age + "<br>";    
       str += "性别：" + gender;    
       document.write(str);    
}    
//-->    
</script>

代码执行效果如下:    
姓名: kobe bryant    
年龄：28    
性别：boy

二、JavaScript HTML DOM

### 2.1 JavaScript HTML DOM简介

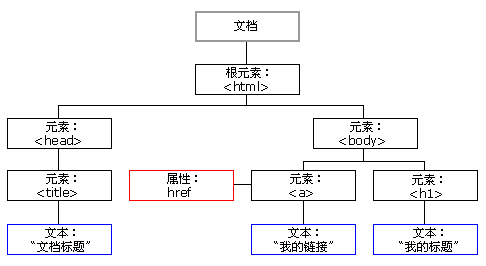
**通过 HTML DOM，可访问 JavaScript HTML 文档的所有元素。**

#### HTML DOM （文档对象模型）

当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model）。

HTML DOM 模型被构造为对象的树。

#### HTML DOM 树



通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML。

* JavaScript 能够改变页面中的所有 HTML 元素
* JavaScript 能够改变页面中的所有 HTML 属性
* JavaScript 能够改变页面中的所有 CSS 样式
* JavaScript 能够对页面中的所有事件做出反应

#### 查找 HTML 元素

通常，通过 JavaScript，您需要操作 HTML 元素。

为了做到这件事情，您必须首先找到该元素。有三种方法来做这件事：

* 通过 id 找到 HTML 元素

var x=document.getElementById("intro");

* 通过标签名找到 HTML 元素

本例查找 id="main" 的元素，然后查找 "main" 中的所有 <p> 元素：

var x=document.getElementById("main");

var y=x.getElementsByTagName("p");

* 通过类名找到 HTML 元素

**提示：**通过类名查找 HTML 元素在 IE 5,6,7,8 中无效。

### 2.2 JavaScript HTML DOM - 改变 HTML

**HTML DOM 允许 JavaScript 改变 HTML 元素的内容。**

#### Attr 对象

在 HTML DOM 中，**Attr** 对象表示 HTML 属性。

HTML 属性始终属于 HTML 元素。

#### NamedNodeMap 对象

在 HTML DOM 中，**NamedNodeMap** 对象表示元素属性节点的无序集合。

NamedNodeMap 中的节点可通过名称或索引（数字）来访问。

#### 属性和方法

|  |  |
| --- | --- |
| **属性 / 方法** | **描述** |
| [attr.isId](http://www.w3school.com.cn/jsref/prop_attr_isid.asp) | 如果属性是 id 类型，则返回 true，否则返回 false。 |
| [attr.name](http://www.w3school.com.cn/jsref/prop_attr_name.asp) | 返回属性的名称。 |
| [attr.value](http://www.w3school.com.cn/jsref/prop_attr_value.asp) | 设置或返回属性的值。 |
| [attr.specified](http://www.w3school.com.cn/jsref/prop_attr_specified.asp) | 如果已指定属性，则返回 true，否则返回 false。 |
| [nodemap.getNamedItem()](http://www.w3school.com.cn/jsref/met_namednodemap_getnameditem.asp) | 从 NamedNodeMap 返回指定的属性节点。 |
| [nodemap.item()](http://www.w3school.com.cn/jsref/met_namednodemap_item.asp) | 返回 NamedNodeMap 中位于指定下标的节点。 |
| [nodemap.length](http://www.w3school.com.cn/jsref/prop_namednodemap_length.asp) | 返回 NamedNodeMap 中的节点数。 |
| [nodemap.removeNamedItem()](http://www.w3school.com.cn/jsref/met_namednodemap_removenameditem.asp) | 移除指定的属性节点。 |
| [nodemap.setNamedItem()](http://www.w3school.com.cn/jsref/met_namednodemap_setnameditem.asp) | 设置指定的属性节点（通过名称）。 |

#### DOM 4 警告！

在 W3C DOM Core 中，Attr (attribute) 对象从 Node 对象继承所有属性和方法。

在 DOM 4 中，Attr 对象不再从 Node 继承。

为了保证未来的代码安全，您应该避免在属性对象上使用节点对象的属性和方法：

|  |  |
| --- | --- |
| **属性 / 方法** | **避免的理由** |
| attr.appendChild() | 属性没有子节点。 |
| attr.attributes | 属性没有属性。 |
| attr.baseURI | 使用 document.baseURI 代替。 |
| attr.childNodes | 属性没有子节点。 |
| attr.cloneNode() | 使用 attr.value 代替。 |
| attr.firstChild | 属性没有子节点。 |
| attr.hasAttributes() | 属性没有属性。 |
| attr.hasChildNodes | 属性没有子节点。 |
| attr.insertBefore() | 属性没有子节点。 |
| attr.isEqualNode() | 没有意义。 |
| attr.isSameNode() | 没有意义。 |
| attr.isSupported() | 始终为 true。 |
| attr.lastChild | 属性没有子节点。 |
| attr.nextSibling | 属性没有同级节点。 |
| attr.nodeName | 使用 attr.name 代替。 |
| attr.nodeType | 始终为 2 (ATTRIBUTE\_NODE)。 |
| attr.nodeValue | 使用 attr.value 代替。 |
| attr.normalize() | 属性无法被正常化。 |
| attr.ownerDocument | 始终是您的 HTML 文档。 |
| attr.ownerElement | 这是您用来访问该属性的 HTML 元素。 |
| attr.parentNode | 这是您用来访问该属性的 HTML 元素。 |
| attr.previousSibling | 属性没有同级节点。 |
| attr.removeChild | 属性没有子节点。 |
| attr.replaceChild | 属性没有子节点。 |
| attr.textContent | 使用 attr.value 代替。 |

#### 改变 HTML 输出流

在 JavaScript 中，document.write() 可用于直接向 HTML 输出流写内容。

<!DOCTYPE html>

<html>

<body>

<script>

document.write(Date());

</script>

</body>

</html>

**提示：**绝不要使用在文档加载之后使用 document.write()。这会覆盖该文档。

#### 改变 HTML 内容

修改 HTML 内容的最简单的方法时使用 innerHTML 属性。

document.getElementById(*id*).innerHTML=*new HTML*

#### 改变 HTML 属性

改变 HTML 元素的属性。

document.getElementById(*id*).attribute=*new value*

### 2.3 JavaScript HTML DOM - 改变 CSS

**HTML DOM 允许 JavaScript 改变 HTML 元素的样式。**

#### 改变 HTML 样式

语法：

document.getElementById(*id*).style.property=*new style*

### 2.4 JavaScript HTML DOM 事件

**HTML DOM 使 JavaScript 有能力对 HTML 事件做出反应。**

#### 对事件做出反应

我们可以在事件发生时执行 JavaScript，比如当用户在 HTML 元素上点击时。

如需在用户点击某个元素时执行代码，请向一个 HTML 事件属性添加 JavaScript 代码：

onclick=*JavaScript*

HTML 事件的例子：

* 当用户点击鼠标时
* 当网页已加载时
* 当图像已加载时
* 当鼠标移动到元素上时
* 当输入字段被改变时
* 当提交 HTML 表单时
* 当用户触发按键时

**例子 1**

在本例中，当用户在 <h1> 元素上点击时，会改变其内容：

<h1 onclick="this.innerHTML='谢谢!'">请点击该文本</h1>

**例子 2**

本例从事件处理器调用一个函数：

<!DOCTYPE html>

<html>

<head>

<script>

function changetext(id)

{

id.innerHTML="谢谢!";

}

</script>

</head>

<body>

<h1 onclick="changetext(this)">请点击该文本</h1>

</body>

</html>

#### HTML 事件属性

如需向 HTML 元素分配 事件，您可以使用事件属性。

**实例**

向 button 元素分配 onclick 事件：

<button onclick="displayDate()">点击这里</button>

#### 使用 HTML DOM 来分配事件

HTML DOM 允许您通过使用 JavaScript 来向 HTML 元素分配事件：

**实例**

向 button 元素分配 onclick 事件：

<script>

document.getElementById("myBtn").onclick=function(){displayDate()};

</script>

#### onload 和 onunload 事件

onload 和 onunload 事件会在用户进入或离开页面时被触发。

onload 事件可用于检测访问者的浏览器类型和浏览器版本，并基于这些信息来加载网页的正确版本。

onload 和 onunload 事件可用于处理 cookie。

<body onload="checkCookies()">

#### onchange 事件

onchange 事件常结合对输入字段的验证来使用。

下面是一个如何使用 onchange 的例子。当用户改变输入字段的内容时，会调用 upperCase() 函数。

**实例**

<input type="text" id="fname" onchange="upperCase()">

#### onmouseover 和 onmouseout 事件

onmouseover 和 onmouseout 事件可用于在用户的鼠标移至 HTML 元素上方或移出元素时触发函数。

<!DOCTYPE html>

<html>

<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)" style="background-color:green;width:120px;height:20px;padding:40px;color:#ffffff;">把鼠标移到上面</div>

<script>

function mOver(obj)

{

obj.innerHTML="谢谢"

}

function mOut(obj)

{

obj.innerHTML="把鼠标移到上面"

}

</script>

</body>

</html>

#### onmousedown、onmouseup 以及 onclick 事件

onmousedown, onmouseup 以及 onclick 构成了鼠标点击事件的所有部分。首先当点击鼠标按钮时，会触发 onmousedown 事件，当释放鼠标按钮时，会触发 onmouseup 事件，最后，当完成鼠标点击时，会触发 onclick 事件。

#### Onmouseover、onmouseout事件

Onmouseover当鼠标移动到元素上触发、onmouseout当鼠标离开元素上触发事件。

#### Event 对象

Event 对象代表事件的状态，比如事件在其中发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态。

事件通常与函数结合使用，函数不会在事件发生前被执行！

#### 事件句柄　(Event Handlers)

HTML 4.0 的新特性之一是能够使 HTML 事件触发浏览器中的行为，比如当用户点击某个 HTML 元素时启动一段 JavaScript。下面是一个属性列表，可将之插入 HTML 标签以定义事件的行为。

|  |  |
| --- | --- |
| **属性** | **此事件发生在何时...** |
| [onabort](http://www.w3school.com.cn/jsref/event_onabort.asp) | 图像的加载被中断。 |
| [onblur](http://www.w3school.com.cn/jsref/event_onblur.asp) | 元素失去焦点。 |
| [onchange](http://www.w3school.com.cn/jsref/event_onchange.asp) | 域的内容被改变。 |
| [onclick](http://www.w3school.com.cn/jsref/event_onclick.asp) | 当用户点击某个对象时调用的事件句柄。 |
| [ondblclick](http://www.w3school.com.cn/jsref/event_ondblclick.asp) | 当用户双击某个对象时调用的事件句柄。 |
| [onerror](http://www.w3school.com.cn/jsref/event_onerror.asp) | 在加载文档或图像时发生错误。 |
| [onfocus](http://www.w3school.com.cn/jsref/event_onfocus.asp) | 元素获得焦点。 |
| [onkeydown](http://www.w3school.com.cn/jsref/event_onkeydown.asp) | 某个键盘按键被按下。 |
| [onkeypress](http://www.w3school.com.cn/jsref/event_onkeypress.asp) | 某个键盘按键被按下并松开。 |
| [onkeyup](http://www.w3school.com.cn/jsref/event_onkeyup.asp) | 某个键盘按键被松开。 |
| [onload](http://www.w3school.com.cn/jsref/event_onload.asp) | 一张页面或一幅图像完成加载。 |
| [onmousedown](http://www.w3school.com.cn/jsref/event_onmousedown.asp) | 鼠标按钮被按下。 |
| [onmousemove](http://www.w3school.com.cn/jsref/event_onmousemove.asp) | 鼠标被移动。 |
| [onmouseout](http://www.w3school.com.cn/jsref/event_onmouseout.asp) | 鼠标从某元素移开。 |
| [onmouseover](http://www.w3school.com.cn/jsref/event_onmouseover.asp) | 鼠标移到某元素之上。 |
| [onmouseup](http://www.w3school.com.cn/jsref/event_onmouseup.asp) | 鼠标按键被松开。 |
| [onreset](http://www.w3school.com.cn/jsref/event_onreset.asp) | 重置按钮被点击。 |
| [onresize](http://www.w3school.com.cn/jsref/event_onresize.asp) | 窗口或框架被重新调整大小。 |
| [onselect](http://www.w3school.com.cn/jsref/event_onselect.asp) | 文本被选中。 |
| [onsubmit](http://www.w3school.com.cn/jsref/event_onsubmit.asp) | 确认按钮被点击。 |
| [onunload](http://www.w3school.com.cn/jsref/event_onunload.asp) | 用户退出页面。 |

#### 鼠标 / 键盘属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [altKey](http://www.w3school.com.cn/jsref/event_altkey.asp) | 返回当事件被触发时，"ALT" 是否被按下。 |
| [button](http://www.w3school.com.cn/jsref/event_button.asp) | 返回当事件被触发时，哪个鼠标按钮被点击。 |
| [clientX](http://www.w3school.com.cn/jsref/event_clientx.asp) | 返回当事件被触发时，鼠标指针的水平坐标。 |
| [clientY](http://www.w3school.com.cn/jsref/event_clienty.asp) | 返回当事件被触发时，鼠标指针的垂直坐标。 |
| [ctrlKey](http://www.w3school.com.cn/jsref/event_ctrlkey.asp) | 返回当事件被触发时，"CTRL" 键是否被按下。 |
| [metaKey](http://www.w3school.com.cn/jsref/event_metakey.asp) | 返回当事件被触发时，"meta" 键是否被按下。 |
| [relatedTarget](http://www.w3school.com.cn/jsref/event_relatedtarget.asp) | 返回与事件的目标节点相关的节点。 |
| [screenX](http://www.w3school.com.cn/jsref/event_screenx.asp) | 返回当某个事件被触发时，鼠标指针的水平坐标。 |
| [screenY](http://www.w3school.com.cn/jsref/event_screeny.asp) | 返回当某个事件被触发时，鼠标指针的垂直坐标。 |
| [shiftKey](http://www.w3school.com.cn/jsref/event_shiftkey.asp) | 返回当事件被触发时，"SHIFT" 键是否被按下。 |

#### IE 属性

除了上面的鼠标/事件属性，IE 浏览器还支持下面的属性：

|  |  |
| --- | --- |
| **属性** | **描述** |
| cancelBubble | 如果事件句柄想阻止事件传播到包容对象，必须把该属性设为 true。 |
| fromElement | 对于 mouseover 和 mouseout 事件，fromElement 引用移出鼠标的元素。 |
| keyCode | 对于 keypress 事件，该属性声明了被敲击的键生成的 Unicode 字符码。对于 keydown 和 keyup 事件，它指定了被敲击的键的虚拟键盘码。虚拟键盘码可能和使用的键盘的布局相关。 |
| offsetX,offsetY | 发生事件的地点在事件源元素的坐标系统中的 x 坐标和 y 坐标。 |
| returnValue | 如果设置了该属性，它的值比事件句柄的返回值优先级高。把这个属性设置为 fasle，可以取消发生事件的源元素的默认动作。 |
| srcElement | 对于生成事件的 Window 对象、Document 对象或 Element 对象的引用。 |
| toElement | 对于 mouseover 和 mouseout 事件，该属性引用移入鼠标的元素。 |
| x,y | 事件发生的位置的 x 坐标和 y 坐标，它们相对于用CSS动态定位的最内层包容元素。 |

#### 标准 Event 属性

下面列出了 2 级 DOM 事件标准定义的属性。

|  |  |
| --- | --- |
| **属性** | **描述** |
| [bubbles](http://www.w3school.com.cn/jsref/event_bubbles.asp) | 返回布尔值，指示事件是否是起泡事件类型。 |
| [cancelable](http://www.w3school.com.cn/jsref/event_cancelable.asp) | 返回布尔值，指示事件是否可拥可取消的默认动作。 |
| [currentTarget](http://www.w3school.com.cn/jsref/event_currenttarget.asp) | 返回其事件监听器触发该事件的元素。 |
| [eventPhase](http://www.w3school.com.cn/jsref/event_eventphase.asp) | 返回事件传播的当前阶段。 |
| [target](http://www.w3school.com.cn/jsref/event_target.asp) | 返回触发此事件的元素（事件的目标节点）。 |
| [timeStamp](http://www.w3school.com.cn/jsref/event_timestamp.asp) | 返回事件生成的日期和时间。 |
| [type](http://www.w3school.com.cn/jsref/event_type.asp) | 返回当前 Event 对象表示的事件的名称。 |

## 标准 Event 方法

下面列出了 2 级 DOM 事件标准定义的方法。IE 的事件模型不支持这些方法：

|  |  |
| --- | --- |
| **方法** | **描述** |
| [initEvent()](http://www.w3school.com.cn/jsref/event_initevent.asp) | 初始化新创建的 Event 对象的属性。 |
| [preventDefault()](http://www.w3school.com.cn/jsref/event_preventdefault.asp) | 通知浏览器不要执行与事件关联的默认动作。 |
| [stopPropagation()](http://www.w3school.com.cn/jsref/event_stoppropagation.asp) | 不再派发事件。 |

### 2.5 JavaScript HTML DOM 元素（节点）

**HTML DOM Element 对象**

#### HTML DOM 节点

在 HTML DOM （文档对象模型）中，每个部分都是节点：

* 文档本身是文档节点
* 所有 HTML 元素是元素节点
* 所有 HTML 属性是属性节点
* HTML 元素内的文本是文本节点
* 注释是注释节点

#### Element 对象

在 HTML DOM 中，Element 对象表示 HTML 元素。

Element 对象可以拥有类型为元素节点、文本节点、注释节点的子节点。

NodeList 对象表示节点列表，比如 HTML 元素的子节点集合。

元素也可以拥有属性。属性是属性节点（参见下一节）。

#### 属性和方法

下面的属性和方法可用于所有 HTML 元素上：

|  |  |
| --- | --- |
| **属性 / 方法** | **描述** |
| [element.accessKey](http://www.w3school.com.cn/jsref/prop_html_accesskey.asp) | 设置或返回元素的快捷键。 |
| [element.appendChild()](http://www.w3school.com.cn/jsref/met_node_appendchild.asp) | 向元素添加新的子节点，作为最后一个子节点。 |
| [element.attributes](http://www.w3school.com.cn/jsref/prop_node_attributes.asp) | 返回元素属性的 NamedNodeMap。 |
| [element.childNodes](http://www.w3school.com.cn/jsref/prop_node_childnodes.asp) | 返回元素子节点的 NodeList。 |
| [element.className](http://www.w3school.com.cn/jsref/prop_html_classname.asp) | 设置或返回元素的 class 属性。 |
| element.clientHeight | 返回元素的可见高度。 |
| element.clientWidth | 返回元素的可见宽度。 |
| [element.cloneNode()](http://www.w3school.com.cn/jsref/met_node_clonenode.asp) | 克隆元素。 |
| [element.compareDocumentPosition()](http://www.w3school.com.cn/jsref/met_node_comparedocumentposition.asp) | 比较两个元素的文档位置。 |
| [element.contentEditable](http://www.w3school.com.cn/jsref/prop_html_contenteditable.asp) | 设置或返回元素的文本方向。 |
| [element.dir](http://www.w3school.com.cn/jsref/prop_html_dir.asp) | 设置或返回元素的文本方向。 |
| [element.firstChild](http://www.w3school.com.cn/jsref/prop_node_firstchild.asp) | 返回元素的首个子。 |
| [element.getAttribute()](http://www.w3school.com.cn/jsref/met_element_getattribute.asp) | 返回元素节点的指定属性值。 |
| [element.getAttributeNode()](http://www.w3school.com.cn/jsref/met_element_getattributenode.asp) | 返回指定的属性节点。 |
| [element.getElementsByTagName()](http://www.w3school.com.cn/jsref/met_element_getelementsbytagname.asp) | 返回拥有指定标签名的所有子元素的集合。 |
| element.getFeature() | 返回实现了指定特性的 API 的某个对象。 |
| element.getUserData() | 返回关联元素上键的对象。 |
| [element.hasAttribute()](http://www.w3school.com.cn/jsref/met_element_hasattribute.asp) | 如果元素拥有指定属性，则返回true，否则返回 false。 |
| [element.hasAttributes()](http://www.w3school.com.cn/jsref/met_node_hasattributes.asp) | 如果元素拥有属性，则返回 true，否则返回 false。 |
| [element.hasChildNodes()](http://www.w3school.com.cn/jsref/met_node_haschildnodes.asp) | 如果元素拥有子节点，则返回 true，否则 false。 |
| [element.id](http://www.w3school.com.cn/jsref/prop_html_id.asp) | 设置或返回元素的 id。 |
| [element.innerHTML](http://www.w3school.com.cn/jsref/prop_html_innerhtml.asp) | 设置或返回元素的内容。 |
| [element.insertBefore()](http://www.w3school.com.cn/jsref/met_node_insertbefore.asp) | 在指定的已有的子节点之前插入新节点。 |
| [element.isContentEditable](http://www.w3school.com.cn/jsref/prop_html_iscontenteditable.asp) | 设置或返回元素的内容。 |
| [element.isDefaultNamespace()](http://www.w3school.com.cn/jsref/met_node_isdefaultnamespace.asp) | 如果指定的 namespaceURI 是默认的，则返回 true，否则返回 false。 |
| [element.isEqualNode()](http://www.w3school.com.cn/jsref/met_node_isequalnode.asp) | 检查两个元素是否相等。 |
| [element.isSameNode()](http://www.w3school.com.cn/jsref/met_node_issamenode.asp) | 检查两个元素是否是相同的节点。 |
| [element.isSupported()](http://www.w3school.com.cn/jsref/met_node_issupported.asp) | 如果元素支持指定特性，则返回 true。 |
| [element.lang](http://www.w3school.com.cn/jsref/prop_html_lang.asp) | 设置或返回元素的语言代码。 |
| [element.lastChild](http://www.w3school.com.cn/jsref/prop_node_lastchild.asp) | 返回元素的最后一个子元素。 |
| [element.namespaceURI](http://www.w3school.com.cn/jsref/prop_node_namespaceuri.asp) | 返回元素的 namespace URI。 |
| [element.nextSibling](http://www.w3school.com.cn/jsref/prop_node_nextsibling.asp) | 返回位于相同节点树层级的下一个节点。 |
| [element.nodeName](http://www.w3school.com.cn/jsref/prop_node_nodename.asp) | 返回元素的名称。 |
| [element.nodeType](http://www.w3school.com.cn/jsref/prop_node_nodetype.asp) | 返回元素的节点类型。 |
| [element.nodeValue](http://www.w3school.com.cn/jsref/prop_node_nodevalue.asp) | 设置或返回元素值。 |
| [element.normalize()](http://www.w3school.com.cn/jsref/met_node_normalize.asp) | 合并元素中相邻的文本节点，并移除空的文本节点。 |
| element.offsetHeight | 返回元素的高度。 |
| element.offsetWidth | 返回元素的宽度。 |
| element.offsetLeft | 返回元素的水平偏移位置。 |
| element.offsetParent | 返回元素的偏移容器。 |
| element.offsetTop | 返回元素的垂直偏移位置。 |
| [element.ownerDocument](http://www.w3school.com.cn/jsref/prop_node_ownerdocument.asp) | 返回元素的根元素（文档对象）。 |
| [element.parentNode](http://www.w3school.com.cn/jsref/prop_node_parentnode.asp) | 返回元素的父节点。 |
| [element.previousSibling](http://www.w3school.com.cn/jsref/prop_node_previoussibling.asp) | 返回位于相同节点树层级的前一个元素。 |
| [element.removeAttribute()](http://www.w3school.com.cn/jsref/met_element_removeattribute.asp) | 从元素中移除指定属性。 |
| [element.removeAttributeNode()](http://www.w3school.com.cn/jsref/met_element_removeattributenode.asp) | 移除指定的属性节点，并返回被移除的节点。 |
| [element.removeChild()](http://www.w3school.com.cn/jsref/met_node_removechild.asp) | 从元素中移除子节点。 |
| [element.replaceChild()](http://www.w3school.com.cn/jsref/met_node_replacechild.asp) | 替换元素中的子节点。 |
| element.scrollHeight | 返回元素的整体高度。 |
| element.scrollLeft | 返回元素左边缘与视图之间的距离。 |
| element.scrollTop | 返回元素上边缘与视图之间的距离。 |
| element.scrollWidth | 返回元素的整体宽度。 |
| [element.setAttribute()](http://www.w3school.com.cn/jsref/met_element_setattribute.asp) | 把指定属性设置或更改为指定值。 |
| [element.setAttributeNode()](http://www.w3school.com.cn/jsref/met_element_setattributenode.asp) | 设置或更改指定属性节点。 |
| element.setIdAttribute() |  |
| element.setIdAttributeNode() |  |
| element.setUserData() | 把对象关联到元素上的键。 |
| element.style | 设置或返回元素的 style 属性。 |
| [element.tabIndex](http://www.w3school.com.cn/jsref/prop_html_tabindex.asp) | 设置或返回元素的 tab 键控制次序。 |
| [element.tagName](http://www.w3school.com.cn/jsref/prop_element_tagname.asp) | 返回元素的标签名。 |
| [element.textContent](http://www.w3school.com.cn/jsref/prop_node_textcontent.asp) | 设置或返回节点及其后代的文本内容。 |
| [element.title](http://www.w3school.com.cn/jsref/prop_html_title.asp) | 设置或返回元素的 title 属性。 |
| element.toString() | 把元素转换为字符串。 |
| [nodelist.item()](http://www.w3school.com.cn/jsref/met_nodelist_item.asp) | 返回 NodeList 中位于指定下标的节点。 |
| [nodelist.length](http://www.w3school.com.cn/jsref/prop_nodelist_length.asp) | 返回 NodeList 中的节点数。 |

**添加和删除节点（HTML 元素）。**

#### 创建新的 HTML 元素

如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

<div id="div1">

<p id="p1">这是一个段落</p>

<p id="p2">这是另一个段落</p>

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("这是新段落。");

para.appendChild(node);

var element=document.getElementById("div1");

element.appendChild(para);

</script>

例子解释：

这段代码创建新的 <p> 元素：

var para=document.createElement("p");

如需向 <p> 元素添加文本，您必须首先创建文本节点。这段代码创建了一个文本节点：

var node=document.createTextNode("这是新段落。");

然后您必须向 <p> 元素追加这个文本节点：

para.appendChild(node);

最后您必须向一个已有的元素追加这个新元素。

这段代码找到一个已有的元素：

var element=document.getElementById("div1");

这段代码向这个已有的元素追加新元素：

element.appendChild(para);

#### 删除已有的 HTML 元素

如需删除 HTML 元素，您必须首先获得该元素的父元素：

**实例**

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另一个段落。</p>

</div>

<script>

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.removeChild(child);

</script>

例子解释：

这个 HTML 文档含有拥有两个子节点（两个 <p> 元素）的 <div> 元素：

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另一个段落。</p>

</div>

找到 id="div1" 的元素：

var parent=document.getElementById("div1");

找到 id="p1" 的 <p> 元素：

var child=document.getElementById("p1");

从父元素中删除子元素：

parent.removeChild(child);

**提示：**如果能够在不引用父元素的情况下删除某个元素，就太好了。

不过很遗憾。DOM 需要清楚您需要删除的元素，以及它的父元素。

这是常用的解决方案：找到您希望删除的子元素，然后使用其 parentNode 属性来找到父元素：

var child=document.getElementById("p1");

child.parentNode.removeChild(child);

### 2.6 HTML DOM Document 对象

#### Document 对象

每个载入浏览器的 HTML 文档都会成为 Document 对象。

Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。

**提示：**Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问。

#### Document 对象集合

|  |  |
| --- | --- |
| **集合** | **描述** |
| [all[]](http://www.w3school.com.cn/jsref/coll_doc_all.asp) | 提供对文档中所有 HTML 元素的访问。 |
| [anchors[]](http://www.w3school.com.cn/jsref/coll_doc_anchors.asp) | 返回对文档中所有 Anchor 对象的引用。 |
| applets | 返回对文档中所有 Applet 对象的引用。 |
| [forms[]](http://www.w3school.com.cn/jsref/coll_doc_forms.asp) | 返回对文档中所有 Form 对象引用。 |
| [images[]](http://www.w3school.com.cn/jsref/coll_doc_images.asp) | 返回对文档中所有 Image 对象引用。 |
| [links[]](http://www.w3school.com.cn/jsref/coll_doc_links.asp) | 返回对文档中所有 Area 和 Link 对象引用。 |

#### Document 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| body | 提供对 <body> 元素的直接访问。  对于定义了框架集的文档，该属性引用最外层的 <frameset>。 |
| [cookie](http://www.w3school.com.cn/jsref/prop_doc_cookie.asp) | 设置或返回与当前文档有关的所有 cookie。 |
| [domain](http://www.w3school.com.cn/jsref/prop_doc_domain.asp) | 返回当前文档的域名。 |
| [lastModified](http://www.w3school.com.cn/jsref/prop_doc_lastmodified.asp) | 返回文档被最后修改的日期和时间。 |
| [referrer](http://www.w3school.com.cn/jsref/prop_doc_referrer.asp) | 返回载入当前文档的文档的 URL。 |
| [title](http://www.w3school.com.cn/jsref/prop_doc_title.asp) | 返回当前文档的标题。 |
| [URL](http://www.w3school.com.cn/jsref/prop_doc_url.asp) | 返回当前文档的 URL。 |

#### Document 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [close()](http://www.w3school.com.cn/jsref/met_doc_close.asp) | 关闭用 document.open() 方法打开的输出流，并显示选定的数据。 |
| [getElementById()](http://www.w3school.com.cn/jsref/met_doc_getelementbyid.asp) | 返回对拥有指定 id 的第一个对象的引用。 |
| [getElementsByName()](http://www.w3school.com.cn/jsref/met_doc_getelementsbyname.asp) | 返回带有指定名称的对象集合。 |
| [getElementsByTagName()](http://www.w3school.com.cn/jsref/met_doc_getelementsbytagname.asp) | 返回带有指定标签名的对象集合。 |
| [open()](http://www.w3school.com.cn/jsref/met_doc_open.asp) | 打开一个流，以收集来自任何 document.write() 或 document.writeln() 方法的输出。 |
| [write()](http://www.w3school.com.cn/jsref/met_doc_write.asp) | 向文档写 HTML 表达式 或 JavaScript 代码。 |
| [writeln()](http://www.w3school.com.cn/jsref/met_doc_writeln.asp) | 等同于 write() 方法，不同的是在每个表达式之后写一个换行符。 |

#### Document 对象描述

HTMLDocument 接口对 DOM Document 接口进行了扩展，定义 HTML 专用的属性和方法。

很多属性和方法都是 HTMLCollection 对象（实际上是可以用数组或名称索引的只读数组），其中保存了对锚、表单、链接以及其他可脚本元素的引用。

这些集合属性都源自于 0 级 DOM。它们已经被 [Document.getElementsByTagName()](http://www.w3school.com.cn/jsref/met_doc_getelementsbytagname.asp) 所取代，但是仍然常常使用，因为他们很方便。

[write() 方法](http://www.w3school.com.cn/jsref/met_doc_write.asp)值得注意，在文档载入和解析的时候，它允许一个脚本向文档中插入动态生成的内容。

注意，在 1 级 DOM 中，HTMLDocument 定义了一个名为 [getElementById()](http://www.w3school.com.cn/jsref/met_doc_getelementbyid.asp) 的非常有用的方法。在 2 级 DOM 中，该方法已经被转移到了 Document 接口，它现在由 HTMLDocument 继承而不是由它定义了。

三、JavaScript 对象

### 3.1 JavaScript 对象

**JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...**

**此外，JavaScript 允许自定义对象。**

#### JavaScript 对象

JavaScript 提供多个**内建**对象，比如 String、Date、Array 等等。

对象只是带有**属性**和**方法**的特殊数据类型。

#### 访问对象的属性

属性是与对象相关的值。

**语法：**

*objectName*.*propertyName*

#### 访问对象的方法

方法是能够在对象上执行的动作。

**语法：**

*objectName*.*methodName*()

#### 创建 JavaScript 对象

通过 JavaScript，您能够定义并创建自己的对象。

创建新对象有两种不同的方法：

1. 定义并创建对象的实例
2. 使用函数来定义对象，然后创建新的对象实例

##### 创建直接的实例

这个例子创建了对象的一个新实例，并向其添加了四个属性：

person=new Object();

person.firstname="Bill";

person.lastname="Gates";

person.age=56;

person.eyecolor="blue";

替代语法（使用对象 literals）：

person={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};

##### 使用对象构造器

本例使用函数来构造对象：

function person(firstname,lastname,age,eyecolor)

{

this.firstname=firstname;

this.lastname=lastname;

this.age=age;

this.eyecolor=eyecolor;

}

**创建 JavaScript 对象实例**

一旦您有了对象构造器，就可以创建新的对象实例，就像这样：

var myFather=new person("Bill","Gates",56,"blue");

var myMother=new person("Steve","Jobs",48,"green");

#### 把属性添加到 JavaScript 对象

您可以通过为对象赋值，向已有对象添加新属性：

假设 personObj 已存在 - 您可以为其添加这些新属性：firstname、lastname、age 以及 eyecolor：

person.firstname="Bill";

person.lastname="Gates";

person.age=56;

person.eyecolor="blue";

x=person.firstname;

#### 把方法添加到 JavaScript 对象

方法只不过是附加在对象上的函数。

在构造器函数内部定义对象的方法：

function person(firstname,lastname,age,eyecolor)

{

this.firstname=firstname;

this.lastname=lastname;

this.age=age;

this.eyecolor=eyecolor;

this.changeName=changeName;

function changeName(name)

{

this.lastname=name;

}

}

#### JavaScript 类

JavaScript 是面向对象的语言，但 JavaScript 不使用类。

在 JavaScript 中，不会创建类，也不会通过类来创建对象（就像在其他面向对象的语言中那样）。

JavaScript 基于 prototype，而不是基于类的。

#### JavaScript for...in 循环

JavaScript for...in 语句循环遍历对象的属性。

**语法**

for (对象中的变量)

{

要执行的代码

}

**注释：**for...in 循环中的代码块将针对每个属性执行一次。

**实例**

循环遍历对象的属性：

var person={fname:"Bill",lname:"Gates",age:56};

for (x in person)

{

txt=txt + person[x];

}

### 3.2 JavaScript Number 对象

JavaScript 只有一种数字类型。

可以使用也可以不使用小数点来书写数字。

var pi=3.14; // 使用小数点

var x=34; // 不使用小数点

极大或极小的数字可通过科学（指数）计数法来写：

var y=123e5; // 12300000

var z=123e-5; // 0.00123

**创建 Number 对象的语法：**

var myNum=new Number(*value*);

var myNum=Number(*value*);

#### 所有 JavaScript 数字均为 64 位

JavaScript 不是类型语言。与许多其他编程语言不同，JavaScript 不定义不同类型的数字，比如整数、短、长、浮点等等。

JavaScript 中的所有数字都存储为根为 10 的 64 位（8 比特），浮点数。

#### 精度

整数（不使用小数点或指数计数法）最多为 15 位。

小数的最大位数是 17，但是浮点运算并不总是 100% 准确：

var x=0.2+0.1;

#### 八进制和十六进制

如果前缀为 0，则 JavaScript 会把数值常量解释为八进制数，如果前缀为 0 和 "x"，则解释为十六进制数。

var y=0377;

var z=0xFF;

**提示：**绝不要在数字前面写零，除非您需要进行八进制转换。

#### 数字属性和方法

##### 属性：

* MAX VALUE
* MIN VALUE
* NEGATIVE INFINITIVE
* POSITIVE INFINITIVE
* NaN
* prototype
* constructor

##### 方法：

* toExponential()
* toFixed()
* toPrecision()
* toString()
* valueOf()

#### Number 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_number.asp) | 返回对创建此对象的 Number 函数的引用。 |
| [MAX\_VALUE](http://www.w3school.com.cn/jsref/jsref_max_value.asp) | 可表示的最大的数。 |
| [MIN\_VALUE](http://www.w3school.com.cn/jsref/jsref_min_value.asp) | 可表示的最小的数。 |
| [NaN](http://www.w3school.com.cn/jsref/jsref_nan_number.asp) | 非数字值。 |
| [NEGATIVE\_INFINITY](http://www.w3school.com.cn/jsref/jsref_negative_infinity.asp) | 负无穷大，溢出时返回该值。 |
| [POSITIVE\_INFINITY](http://www.w3school.com.cn/jsref/jsref_positive_infinity.asp) | 正无穷大，溢出时返回该值。 |
| prototype | 使您有能力向对象添加属性和方法。 |

#### Number 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [toString](http://www.w3school.com.cn/jsref/jsref_tostring_number.asp) | 把数字转换为字符串，使用指定的基数。 |
| [toLocaleString](http://www.w3school.com.cn/jsref/jsref_tolocalestring_number.asp) | 把数字转换为字符串，使用本地数字格式顺序。 |
| [toFixed](http://www.w3school.com.cn/jsref/jsref_tofixed.asp) | 把数字转换为字符串，结果的小数点后有指定位数的数字。 |
| [toExponential](http://www.w3school.com.cn/jsref/jsref_toexponential.asp) | 把对象的值转换为指数计数法。 |
| [toPrecision](http://www.w3school.com.cn/jsref/jsref_toprecision.asp) | 把数字格式化为指定的长度。 |
| [valueOf](http://www.w3school.com.cn/jsref/jsref_valueof_number.asp) | 返回一个 Number 对象的基本数字值。 |

### 2.3 JavaScript 字符串(String)对象

**String 对象用于处理已有的字符块。**

#### 创建 String 对象的语法：

new String(*s*);

String(*s*);

#### String 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| constructor | 对创建该对象的函数的引用 |
| [length](http://www.w3school.com.cn/jsref/jsref_length_string.asp) | 字符串的长度 |
| prototype | 允许您向对象添加属性和方法 |

#### String 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [anchor()](http://www.w3school.com.cn/jsref/jsref_anchor.asp) | 创建 HTML 锚。 |
| [big()](http://www.w3school.com.cn/jsref/jsref_big.asp) | 用大号字体显示字符串。 |
| [blink()](http://www.w3school.com.cn/jsref/jsref_blink.asp) | 显示闪动字符串。 |
| [bold()](http://www.w3school.com.cn/jsref/jsref_bold.asp) | 使用粗体显示字符串。 |
| [charAt()](http://www.w3school.com.cn/jsref/jsref_charAt.asp) | 返回在指定位置的字符。 |
| [charCodeAt()](http://www.w3school.com.cn/jsref/jsref_charCodeAt.asp) | 返回在指定的位置的字符的 Unicode 编码。 |
| [concat()](http://www.w3school.com.cn/jsref/jsref_concat_string.asp) | 连接字符串。 |
| [fixed()](http://www.w3school.com.cn/jsref/jsref_fixed.asp) | 以打字机文本显示字符串。 |
| [fontcolor()](http://www.w3school.com.cn/jsref/jsref_fontcolor.asp) | 使用指定的颜色来显示字符串。 |
| [fontsize()](http://www.w3school.com.cn/jsref/jsref_fontsize.asp) | 使用指定的尺寸来显示字符串。 |
| [fromCharCode()](http://www.w3school.com.cn/jsref/jsref_fromCharCode.asp) | 从字符编码创建一个字符串。 |
| [indexOf()](http://www.w3school.com.cn/jsref/jsref_indexOf.asp) | 检索字符串。 |
| [italics()](http://www.w3school.com.cn/jsref/jsref_italics.asp) | 使用斜体显示字符串。 |
| [lastIndexOf()](http://www.w3school.com.cn/jsref/jsref_lastIndexOf.asp) | 从后向前搜索字符串。 |
| [link()](http://www.w3school.com.cn/jsref/jsref_link.asp) | 将字符串显示为链接。 |
| [localeCompare()](http://www.w3school.com.cn/jsref/jsref_localeCompare.asp) | 用本地特定的顺序来比较两个字符串。 |
| [match()](http://www.w3school.com.cn/jsref/jsref_match.asp) | 找到一个或多个正则表达式的匹配。 |
| [replace()](http://www.w3school.com.cn/jsref/jsref_replace.asp) | 替换与正则表达式匹配的子串。 |
| [search()](http://www.w3school.com.cn/jsref/jsref_search.asp) | 检索与正则表达式相匹配的值。 |
| [slice()](http://www.w3school.com.cn/jsref/jsref_slice_string.asp) | 提取字符串的片断，并在新的字符串中返回被提取的部分。 |
| [small()](http://www.w3school.com.cn/jsref/jsref_small.asp) | 使用小字号来显示字符串。 |
| [split()](http://www.w3school.com.cn/jsref/jsref_split.asp) | 把字符串分割为字符串数组。 |
| [strike()](http://www.w3school.com.cn/jsref/jsref_strike.asp) | 使用删除线来显示字符串。 |
| [sub()](http://www.w3school.com.cn/jsref/jsref_sub.asp) | 把字符串显示为下标。 |
| [substr()](http://www.w3school.com.cn/jsref/jsref_substr.asp) | 从起始索引号提取字符串中指定数目的字符。 |
| [substring()](http://www.w3school.com.cn/jsref/jsref_substring.asp) | 提取字符串中两个指定的索引号之间的字符。 |
| [sup()](http://www.w3school.com.cn/jsref/jsref_sup.asp) | 把字符串显示为上标。 |
| [toLocaleLowerCase()](http://www.w3school.com.cn/jsref/jsref_toLocaleLowerCase.asp) | 把字符串转换为小写。 |
| [toLocaleUpperCase()](http://www.w3school.com.cn/jsref/jsref_toLocaleUpperCase.asp) | 把字符串转换为大写。 |
| [toLowerCase()](http://www.w3school.com.cn/jsref/jsref_toLowerCase.asp) | 把字符串转换为小写。 |
| [toUpperCase()](http://www.w3school.com.cn/jsref/jsref_toUpperCase.asp) | 把字符串转换为大写。 |
| toSource() | 代表对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_string.asp) | 返回字符串。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueOf_string.asp) | 返回某个字符串对象的原始值。 |

### 3.4 JavaScript Date（日期）对象

**日期对象用于处理日期和时间。**

#### 创建 Date 对象的语法：

var myDate=new Date()

**注释：**Date 对象会自动把当前日期和时间保存为其初始值。

#### Date 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_date.asp) | 返回对创建此对象的 Date 函数的引用。 |
| [prototype](http://www.w3school.com.cn/jsref/jsref_prototype_date.asp) | 使您有能力向对象添加属性和方法。 |

#### Date 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [Date()](http://www.w3school.com.cn/jsref/jsref_Date.asp) | 返回当日的日期和时间。 |
| [getDate()](http://www.w3school.com.cn/jsref/jsref_getDate.asp) | 从 Date 对象返回一个月中的某一天 (1 ~ 31)。 |
| [getDay()](http://www.w3school.com.cn/jsref/jsref_getDay.asp) | 从 Date 对象返回一周中的某一天 (0 ~ 6)。 |
| [getMonth()](http://www.w3school.com.cn/jsref/jsref_getMonth.asp) | 从 Date 对象返回月份 (0 ~ 11)。 |
| [getFullYear()](http://www.w3school.com.cn/jsref/jsref_getFullYear.asp) | 从 Date 对象以四位数字返回年份。 |
| [getYear()](http://www.w3school.com.cn/jsref/jsref_getYear.asp) | 请使用 getFullYear() 方法代替。 |
| [getHours()](http://www.w3school.com.cn/jsref/jsref_getHours.asp) | 返回 Date 对象的小时 (0 ~ 23)。 |
| [getMinutes()](http://www.w3school.com.cn/jsref/jsref_getMinutes.asp) | 返回 Date 对象的分钟 (0 ~ 59)。 |
| [getSeconds()](http://www.w3school.com.cn/jsref/jsref_getSeconds.asp) | 返回 Date 对象的秒数 (0 ~ 59)。 |
| [getMilliseconds()](http://www.w3school.com.cn/jsref/jsref_getMilliseconds.asp) | 返回 Date 对象的毫秒(0 ~ 999)。 |
| [getTime()](http://www.w3school.com.cn/jsref/jsref_getTime.asp) | 返回 1970 年 1 月 1 日至今的毫秒数。 |
| [getTimezoneOffset()](http://www.w3school.com.cn/jsref/jsref_getTimezoneOffset.asp) | 返回本地时间与格林威治标准时间 (GMT) 的分钟差。 |
| [getUTCDate()](http://www.w3school.com.cn/jsref/jsref_getUTCDate.asp) | 根据世界时从 Date 对象返回月中的一天 (1 ~ 31)。 |
| [getUTCDay()](http://www.w3school.com.cn/jsref/jsref_getUTCDay.asp) | 根据世界时从 Date 对象返回周中的一天 (0 ~ 6)。 |
| [getUTCMonth()](http://www.w3school.com.cn/jsref/jsref_getUTCMonth.asp) | 根据世界时从 Date 对象返回月份 (0 ~ 11)。 |
| [getUTCFullYear()](http://www.w3school.com.cn/jsref/jsref_getUTCFullYear.asp) | 根据世界时从 Date 对象返回四位数的年份。 |
| [getUTCHours()](http://www.w3school.com.cn/jsref/jsref_getUTCHours.asp) | 根据世界时返回 Date 对象的小时 (0 ~ 23)。 |
| [getUTCMinutes()](http://www.w3school.com.cn/jsref/jsref_getUTCMinutes.asp) | 根据世界时返回 Date 对象的分钟 (0 ~ 59)。 |
| [getUTCSeconds()](http://www.w3school.com.cn/jsref/jsref_getUTCSeconds.asp) | 根据世界时返回 Date 对象的秒钟 (0 ~ 59)。 |
| [getUTCMilliseconds()](http://www.w3school.com.cn/jsref/jsref_getUTCMilliseconds.asp) | 根据世界时返回 Date 对象的毫秒(0 ~ 999)。 |
| [parse()](http://www.w3school.com.cn/jsref/jsref_parse.asp) | 返回1970年1月1日午夜到指定日期（字符串）的毫秒数。 |
| [setDate()](http://www.w3school.com.cn/jsref/jsref_setDate.asp) | 设置 Date 对象中月的某一天 (1 ~ 31)。 |
| [setMonth()](http://www.w3school.com.cn/jsref/jsref_setMonth.asp) | 设置 Date 对象中月份 (0 ~ 11)。 |
| [setFullYear()](http://www.w3school.com.cn/jsref/jsref_setFullYear.asp) | 设置 Date 对象中的年份（四位数字）。 |
| [setYear()](http://www.w3school.com.cn/jsref/jsref_setYear.asp) | 请使用 setFullYear() 方法代替。 |
| [setHours()](http://www.w3school.com.cn/jsref/jsref_setHours.asp) | 设置 Date 对象中的小时 (0 ~ 23)。 |
| [setMinutes()](http://www.w3school.com.cn/jsref/jsref_setMinutes.asp) | 设置 Date 对象中的分钟 (0 ~ 59)。 |
| [setSeconds()](http://www.w3school.com.cn/jsref/jsref_setSeconds.asp) | 设置 Date 对象中的秒钟 (0 ~ 59)。 |
| [setMilliseconds()](http://www.w3school.com.cn/jsref/jsref_setMilliseconds.asp) | 设置 Date 对象中的毫秒 (0 ~ 999)。 |
| [setTime()](http://www.w3school.com.cn/jsref/jsref_setTime.asp) | 以毫秒设置 Date 对象。 |
| [setUTCDate()](http://www.w3school.com.cn/jsref/jsref_setUTCDate.asp) | 根据世界时设置 Date 对象中月份的一天 (1 ~ 31)。 |
| [setUTCMonth()](http://www.w3school.com.cn/jsref/jsref_setUTCMonth.asp) | 根据世界时设置 Date 对象中的月份 (0 ~ 11)。 |
| [setUTCFullYear()](http://www.w3school.com.cn/jsref/jsref_setUTCFullYear.asp) | 根据世界时设置 Date 对象中的年份（四位数字）。 |
| [setUTCHours()](http://www.w3school.com.cn/jsref/jsref_setutchours.asp) | 根据世界时设置 Date 对象中的小时 (0 ~ 23)。 |
| [setUTCMinutes()](http://www.w3school.com.cn/jsref/jsref_setUTCMinutes.asp) | 根据世界时设置 Date 对象中的分钟 (0 ~ 59)。 |
| [setUTCSeconds()](http://www.w3school.com.cn/jsref/jsref_setUTCSeconds.asp) | 根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。 |
| [setUTCMilliseconds()](http://www.w3school.com.cn/jsref/jsref_setUTCMilliseconds.asp) | 根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_boolean.asp) | 返回该对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_date.asp) | 把 Date 对象转换为字符串。 |
| [toTimeString()](http://www.w3school.com.cn/jsref/jsref_toTimeString.asp) | 把 Date 对象的时间部分转换为字符串。 |
| [toDateString()](http://www.w3school.com.cn/jsref/jsref_toDateString.asp) | 把 Date 对象的日期部分转换为字符串。 |
| [toGMTString()](http://www.w3school.com.cn/jsref/jsref_toGMTString.asp) | 请使用 toUTCString() 方法代替。 |
| [toUTCString()](http://www.w3school.com.cn/jsref/jsref_toUTCString.asp) | 根据世界时，把 Date 对象转换为字符串。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString.asp) | 根据本地时间格式，把 Date 对象转换为字符串。 |
| [toLocaleTimeString()](http://www.w3school.com.cn/jsref/jsref_toLocaleTimeString.asp) | 根据本地时间格式，把 Date 对象的时间部分转换为字符串。 |
| [toLocaleDateString()](http://www.w3school.com.cn/jsref/jsref_toLocaleDateString.asp) | 根据本地时间格式，把 Date 对象的日期部分转换为字符串。 |
| [UTC()](http://www.w3school.com.cn/jsref/jsref_utc.asp) | 根据世界时返回 1970 年 1 月 1 日 到指定日期的毫秒数。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueOf_date.asp) | 返回 Date 对象的原始值。 |

#### 操作日期

通过使用针对日期对象的方法，我们可以很容易地对日期进行操作。

在下面的例子中，我们为日期对象设置了一个特定的日期 (2008 年 8 月 9 日)：

var myDate=new Date()

myDate.setFullYear(2008,7,9)

**注意：**表示月份的参数介于 0 到 11 之间。也就是说，如果希望把月设置为 8 月，则参数应该是 7。

在下面的例子中，我们将日期对象设置为 5 天后的日期：

var myDate=new Date()

myDate.setDate(myDate.getDate()+5)

**注意：**如果增加天数会改变月份或者年份，那么日期对象会自动完成这种转换。

#### 比较日期

日期对象也可用于比较两个日期。

下面的代码将当前日期与 2008 年 8 月 9 日做了比较：

var myDate=new Date();

myDate.setFullYear(2008,8,9);

var today = new Date();

if (myDate>today)

{

alert("Today is before 9th August 2008");

}

else

{

alert("Today is after 9th August 2008");

}

### 3.5 JavaScript Array（数组）对象

**数组对象的作用是：使用单独的变量名来存储一系列的值。**

#### 创建 Array 对象的语法：

new Array();

new Array(*size*);

new Array(*element0*, *element1*, ..., *elementn*);

#### Array 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_array.asp) | 返回对创建此对象的数组函数的引用。 |
| [length](http://www.w3school.com.cn/jsref/jsref_length_array.asp) | 设置或返回数组中元素的数目。 |
| [prototype](http://www.w3school.com.cn/jsref/jsref_prototype_array.asp) | 使您有能力向对象添加属性和方法。 |

#### Array 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [concat()](http://www.w3school.com.cn/jsref/jsref_concat_array.asp) | 连接两个或更多的数组，并返回结果。 |
| [join()](http://www.w3school.com.cn/jsref/jsref_join.asp) | 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。 |
| [pop()](http://www.w3school.com.cn/jsref/jsref_pop.asp) | 删除并返回数组的最后一个元素 |
| [push()](http://www.w3school.com.cn/jsref/jsref_push.asp) | 向数组的末尾添加一个或更多元素，并返回新的长度。 |
| [reverse()](http://www.w3school.com.cn/jsref/jsref_reverse.asp) | 颠倒数组中元素的顺序。 |
| [shift()](http://www.w3school.com.cn/jsref/jsref_shift.asp) | 删除并返回数组的第一个元素 |
| [slice()](http://www.w3school.com.cn/jsref/jsref_slice_array.asp) | 从某个已有的数组返回选定的元素 |
| [sort()](http://www.w3school.com.cn/jsref/jsref_sort.asp) | 对数组的元素进行排序 |
| [splice()](http://www.w3school.com.cn/jsref/jsref_splice.asp) | 删除元素，并向数组添加新元素。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_array.asp) | 返回该对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_array.asp) | 把数组转换为字符串，并返回结果。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString_array.asp) | 把数组转换为本地数组，并返回结果。 |
| [unshift()](http://www.w3school.com.cn/jsref/jsref_unshift.asp) | 向数组的开头添加一个或更多元素，并返回新的长度。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_array.asp) | 返回数组对象的原始值 |

[**文字数组 - sort()**](http://www.w3school.com.cn/tiy/t.asp?f=jseg_sort_1)

如何使用 sort() 方法从字面上对数组进行排序。

arr.sort();

[**数字数组 - sort()**](http://www.w3school.com.cn/tiy/t.asp?f=jseg_sort_2)

如何使用 sort() 方法从数值上对数组进行排序。

arr.sort(sortNumber);

### 3.6 JavaScript Boolean（逻辑）对象

**Boolean（逻辑）对象用于将非逻辑值转换为逻辑值（true 或者 false）。**

Boolean 对象表示两个值："true" 或 "false"。

#### 创建 Boolean 对象的语法：

new Boolean(*value*); //构造函数

Boolean(*value*); //转换函数

**注释：**如果省略 value 参数，或者设置为 0、-0、null、""、false、undefined 或 NaN，则该对象设置为 false。否则设置为 true（即使 value 参数是字符串 "false"）。

下面的所有的代码行均会创建初始值为 false 的 Boolean 对象。

var myBoolean=new Boolean();

var myBoolean=new Boolean(0);

var myBoolean=new Boolean(null);

var myBoolean=new Boolean("");

var myBoolean=new Boolean(false);

var myBoolean=new Boolean(NaN);

下面的所有的代码行均会创初始值为 true 的 Boolean 对象：

var myBoolean=new Boolean(1);

var myBoolean=new Boolean(true);

var myBoolean=new Boolean("true");

var myBoolean=new Boolean("false");

var myBoolean=new Boolean("Bill Gates");

#### Boolean 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_boolean.asp) | 返回对创建此对象的 Boolean 函数的引用 |
| [prototype](http://www.w3school.com.cn/jsref/jsref_prototype_boolean.asp) | 使您有能力向对象添加属性和方法。 |

#### Boolean 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_boolean.asp) | 返回该对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_boolean.asp) | 把逻辑值转换为字符串，并返回结果。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueOf_boolean.asp) | 返回 Boolean 对象的原始值。 |

### 3.7 JavaScript Math（算数）对象

**Math（算数）对象的作用是：执行常见的算数任务。**

#### 使用 Math 的属性和方法的语法：

var pi\_value=Math.PI;

var sqrt\_value=Math.sqrt(15);

**注释：**Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像 Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。

#### Math 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [E](http://www.w3school.com.cn/jsref/jsref_e.asp) | 返回算术常量 e，即自然对数的底数（约等于2.718）。 |
| [LN2](http://www.w3school.com.cn/jsref/jsref_ln2.asp) | 返回 2 的自然对数（约等于0.693）。 |
| [LN10](http://www.w3school.com.cn/jsref/jsref_ln10.asp) | 返回 10 的自然对数（约等于2.302）。 |
| [LOG2E](http://www.w3school.com.cn/jsref/jsref_log2e.asp) | 返回以 2 为底的 e 的对数（约等于 1.414）。 |
| [LOG10E](http://www.w3school.com.cn/jsref/jsref_log10e.asp) | 返回以 10 为底的 e 的对数（约等于0.434）。 |
| [PI](http://www.w3school.com.cn/jsref/jsref_pi.asp) | 返回圆周率（约等于3.14159）。 |
| [SQRT1\_2](http://www.w3school.com.cn/jsref/jsref_sqrt1_2.asp) | 返回返回 2 的平方根的倒数（约等于 0.707）。 |
| [SQRT2](http://www.w3school.com.cn/jsref/jsref_sqrt2.asp) | 返回 2 的平方根（约等于 1.414）。 |

#### Math 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [abs(x)](http://www.w3school.com.cn/jsref/jsref_abs.asp) | 返回数的绝对值。 |
| [acos(x)](http://www.w3school.com.cn/jsref/jsref_acos.asp) | 返回数的反余弦值。 |
| [asin(x)](http://www.w3school.com.cn/jsref/jsref_asin.asp) | 返回数的反正弦值。 |
| [atan(x)](http://www.w3school.com.cn/jsref/jsref_atan.asp) | 以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。 |
| [atan2(y,x)](http://www.w3school.com.cn/jsref/jsref_atan2.asp) | 返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。 |
| [ceil(x)](http://www.w3school.com.cn/jsref/jsref_ceil.asp) | 对数进行上舍入。 |
| [cos(x)](http://www.w3school.com.cn/jsref/jsref_cos.asp) | 返回数的余弦。 |
| [exp(x)](http://www.w3school.com.cn/jsref/jsref_exp.asp) | 返回 e 的指数。 |
| [floor(x)](http://www.w3school.com.cn/jsref/jsref_floor.asp) | 对数进行下舍入。 |
| [log(x)](http://www.w3school.com.cn/jsref/jsref_log.asp) | 返回数的自然对数（底为e）。 |
| [max(x,y)](http://www.w3school.com.cn/jsref/jsref_max.asp) | 返回 x 和 y 中的最高值。 |
| [min(x,y)](http://www.w3school.com.cn/jsref/jsref_min.asp) | 返回 x 和 y 中的最低值。 |
| [pow(x,y)](http://www.w3school.com.cn/jsref/jsref_pow.asp) | 返回 x 的 y 次幂。 |
| [random()](http://www.w3school.com.cn/jsref/jsref_random.asp) | 返回 0 ~ 1 之间的随机数。 |
| [round(x)](http://www.w3school.com.cn/jsref/jsref_round.asp) | 把数四舍五入为最接近的整数。 |
| [sin(x)](http://www.w3school.com.cn/jsref/jsref_sin.asp) | 返回数的正弦。 |
| [sqrt(x)](http://www.w3school.com.cn/jsref/jsref_sqrt.asp) | 返回数的平方根。 |
| [tan(x)](http://www.w3school.com.cn/jsref/jsref_tan.asp) | 返回角的正切。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_math.asp) | 返回该对象的源代码。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_math.asp) | 返回 Math 对象的原始值。 |

### 3.8 JavaScript RegExp 对象

**RegExp 对象用于规定在文本中检索的内容。**

RegExp 是正则表达式的缩写。

当您检索某个文本时，可以使用一种模式来描述要检索的内容。RegExp 就是这种模式。

简单的模式可以是一个单独的字符。

更复杂的模式包括了更多的字符，并可用于解析、格式检查、替换等等。

您可以规定字符串中的检索位置，以及要检索的字符类型，等等。

#### 定义 RegExp

RegExp 对象用于存储检索模式。

通过 new 关键词来定义 RegExp 对象。以下代码定义了名为 patt1 的 RegExp 对象，其模式是 "e"：

var patt1=new RegExp("e");

当您使用该 RegExp 对象在一个字符串中检索时，将寻找的是字符 "e"。

#### RegExp 对象的方法

RegExp 对象有 3 个方法：test()、exec() 以及 compile()。

##### test()

test() 方法检索字符串中的指定值。返回值是 true 或 false。

**例子：**

var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

##### exec()

exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。

**例子：**

var patt1=new RegExp("e");

document.write(patt1.exec("The best things in life are free"));

**例子：**

您可以向 RegExp 对象添加第二个参数，以设定检索。例如，如果需要找到所有某个字符的所有存在，则可以使用 "g" 参数 ("global")。

如需关于如何修改搜索模式的完整信息，请访问我们的 [RegExp 对象参考手册](http://www.w3school.com.cn/jsref/jsref_obj_regexp.asp)。

在使用 "g" 参数时，exec() 的工作原理如下：

* 找到第一个 "e"，并存储其位置
* 如果再次运行 exec()，则从存储的位置开始检索，并找到下一个 "e"，并存储其位置

var patt1=new RegExp("e","g");

do

{

result=patt1.exec("The best things in life are free");

document.write(result);

}

while (result!=null)

由于这个字符串中 6 个 "e" 字母，代码的输出将是：

eeeeeenull

##### compile()

compile() 方法用于改变 RegExp。

compile() 既可以改变检索模式，也可以添加或删除第二个参数。

**例子：**

var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

patt1.compile("d");

document.write(patt1.test("The best things in life are free"));

由于字符串中存在 "e"，而没有 "d"，以上代码的输出是：

truefalse

#### 直接量语法

/pattern/attributes

#### 创建 RegExp 对象的语法：

new RegExp(*pattern*, *attributes*);

#### 参数

参数 *pattern* 是一个字符串，指定了正则表达式的模式或其他正则表达式。

参数 *attributes* 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。ECMAScript 标准化之前，不支持 m 属性。如果 *pattern* 是正则表达式，而不是字符串，则必须省略该参数。

#### 返回值

一个新的 RegExp 对象，具有指定的模式和标志。如果参数 *pattern* 是正则表达式而不是字符串，那么 RegExp() 构造函数将用与指定的 RegExp 相同的模式和标志创建一个新的 RegExp 对象。

如果不用 new 运算符，而将 RegExp() 作为函数调用，那么它的行为与用 new 运算符调用时一样，只是当 *pattern* 是正则表达式时，它只返回 *pattern*，而不再创建一个新的 RegExp 对象。

#### 抛出

SyntaxError - 如果 *pattern* 不是合法的正则表达式，或 *attributes* 含有 "g"、"i" 和 "m" 之外的字符，抛出该异常。

TypeError - 如果 *pattern* 是 RegExp 对象，但没有省略 *attributes* 参数，抛出该异常。

#### 修饰符

|  |  |
| --- | --- |
| **修饰符** | **描述** |
| [i](http://www.w3school.com.cn/jsref/jsref_regexp_i.asp) | 执行对大小写不敏感的匹配。 |
| [g](http://www.w3school.com.cn/jsref/jsref_regexp_g.asp) | 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。 |
| m | 执行多行匹配。 |

#### 方括号

方括号用于查找某个范围内的字符：

|  |  |
| --- | --- |
| **表达式** | **描述** |
| [[abc]](http://www.w3school.com.cn/jsref/jsref_regexp_charset.asp) | 查找方括号之间的任何字符。 |
| [[^abc]](http://www.w3school.com.cn/jsref/jsref_regexp_charset_not.asp) | 查找任何不在方括号之间的字符。 |
| [0-9] | 查找任何从 0 至 9 的数字。 |
| [a-z] | 查找任何从小写 a 到小写 z 的字符。 |
| [A-Z] | 查找任何从大写 A 到大写 Z 的字符。 |
| [A-z] | 查找任何从大写 A 到小写 z 的字符。 |
| [adgk] | 查找给定集合内的任何字符。 |
| [^adgk] | 查找给定集合外的任何字符。 |
| (red|blue|green) | 查找任何指定的选项。 |

#### 元字符

元字符（Metacharacter）是拥有特殊含义的字符：

|  |  |
| --- | --- |
| **元字符** | **描述** |
| [.](http://www.w3school.com.cn/jsref/jsref_regexp_dot.asp) | 查找单个字符，除了换行和行结束符。 |
| [\w](http://www.w3school.com.cn/jsref/jsref_regexp_wordchar.asp) | 查找单词字符。 |
| [\W](http://www.w3school.com.cn/jsref/jsref_regexp_wordchar_non.asp) | 查找非单词字符。 |
| [\d](http://www.w3school.com.cn/jsref/jsref_regexp_digit.asp) | 查找数字。 |
| [\D](http://www.w3school.com.cn/jsref/jsref_regexp_digit_non.asp) | 查找非数字字符。 |
| [\s](http://www.w3school.com.cn/jsref/jsref_regexp_whitespace.asp) | 查找空白字符。 |
| [\S](http://www.w3school.com.cn/jsref/jsref_regexp_whitespace_non.asp) | 查找非空白字符。 |
| [\b](http://www.w3school.com.cn/jsref/jsref_regexp_begin.asp) | 匹配单词边界。 |
| [\B](http://www.w3school.com.cn/jsref/jsref_regexp_begin_not.asp) | 匹配非单词边界。 |
| \0 | 查找 NUL 字符。 |
| [\n](http://www.w3school.com.cn/jsref/jsref_regexp_newline.asp) | 查找换行符。 |
| \f | 查找换页符。 |
| \r | 查找回车符。 |
| \t | 查找制表符。 |
| \v | 查找垂直制表符。 |
| [\xxx](http://www.w3school.com.cn/jsref/jsref_regexp_octal.asp) | 查找以八进制数 xxx 规定的字符。 |
| [\xdd](http://www.w3school.com.cn/jsref/jsref_regexp_hex.asp) | 查找以十六进制数 dd 规定的字符。 |
| [\uxxxx](http://www.w3school.com.cn/jsref/jsref_regexp_unicode_hex.asp) | 查找以十六进制数 xxxx 规定的 Unicode 字符。 |

#### 量词

|  |  |
| --- | --- |
| **量词** | **描述** |
| [n+](http://www.w3school.com.cn/jsref/jsref_regexp_onemore.asp) | 匹配任何包含至少一个 n 的字符串。 |
| [n\*](http://www.w3school.com.cn/jsref/jsref_regexp_zeromore.asp) | 匹配任何包含零个或多个 n 的字符串。 |
| [n?](http://www.w3school.com.cn/jsref/jsref_regexp_zeroone.asp) | 匹配任何包含零个或一个 n 的字符串。 |
| [n{X}](http://www.w3school.com.cn/jsref/jsref_regexp_nx.asp) | 匹配包含 X 个 n 的序列的字符串。 |
| [n{X,Y}](http://www.w3school.com.cn/jsref/jsref_regexp_nxy.asp) | 匹配包含 X 或 Y 个 n 的序列的字符串。 |
| [n{X,}](http://www.w3school.com.cn/jsref/jsref_regexp_nxcomma.asp) | 匹配包含至少 X 个 n 的序列的字符串。 |
| [n$](http://www.w3school.com.cn/jsref/jsref_regexp_ndollar.asp) | 匹配任何结尾为 n 的字符串。 |
| [^n](http://www.w3school.com.cn/jsref/jsref_regexp_ncaret.asp) | 匹配任何开头为 n 的字符串。 |
| [?=n](http://www.w3school.com.cn/jsref/jsref_regexp_nfollow.asp) | 匹配任何其后紧接指定字符串 n 的字符串。 |
| [?!n](http://www.w3school.com.cn/jsref/jsref_regexp_nfollow_not.asp) | 匹配任何其后没有紧接指定字符串 n 的字符串。 |

#### RegExp 对象属性

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **描述** | **FF** | **IE** |
| [global](http://www.w3school.com.cn/jsref/jsref_regexp_global.asp) | RegExp 对象是否具有标志 g。 | 1 | 4 |
| [ignoreCase](http://www.w3school.com.cn/jsref/jsref_regexp_ignorecase.asp) | RegExp 对象是否具有标志 i。 | 1 | 4 |
| [lastIndex](http://www.w3school.com.cn/jsref/jsref_lastindex_regexp.asp) | 一个整数，标示开始下一次匹配的字符位置。 | 1 | 4 |
| [multiline](http://www.w3school.com.cn/jsref/jsref_multiline_regexp.asp) | RegExp 对象是否具有标志 m。 | 1 | 4 |
| [source](http://www.w3school.com.cn/jsref/jsref_source_regexp.asp) | 正则表达式的源文本。 | 1 | 4 |

#### RegExp 对象方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| [compile](http://www.w3school.com.cn/jsref/jsref_regexp_compile.asp) | 编译正则表达式。 | 1 | 4 |
| [exec](http://www.w3school.com.cn/jsref/jsref_exec_regexp.asp) | 检索字符串中指定的值。返回找到的值，并确定其位置。 | 1 | 4 |
| [test](http://www.w3school.com.cn/jsref/jsref_test_regexp.asp) | 检索字符串中指定的值。返回 true 或 false。 | 1 | 4 |

#### 支持正则表达式的 String 对象的方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| [search](http://www.w3school.com.cn/jsref/jsref_search.asp) | 检索与正则表达式相匹配的值。 | 1 | 4 |
| [match](http://www.w3school.com.cn/jsref/jsref_match.asp) | 找到一个或多个正则表达式的匹配。 | 1 | 4 |
| [replace](http://www.w3school.com.cn/jsref/jsref_replace.asp) | 替换与正则表达式匹配的子串。 | 1 | 4 |
| [split](http://www.w3school.com.cn/jsref/jsref_split.asp) | 把字符串分割为字符串数组。 | 1 | 4 |

### 3.9 JavaScript 全局对象

全局属性和函数可用于所有内建的 JavaScript 对象。

#### 顶层函数（全局函数）

|  |  |
| --- | --- |
| **函数** | **描述** |
| [decodeURI()](http://www.w3school.com.cn/jsref/jsref_decodeURI.asp) | 解码某个编码的 URI。 |
| [decodeURIComponent()](http://www.w3school.com.cn/jsref/jsref_decodeURIComponent.asp) | 解码一个编码的 URI 组件。 |
| [encodeURI()](http://www.w3school.com.cn/jsref/jsref_encodeuri.asp) | 把字符串编码为 URI。 |
| [encodeURIComponent()](http://www.w3school.com.cn/jsref/jsref_encodeURIComponent.asp) | 把字符串编码为 URI 组件。 |
| [escape()](http://www.w3school.com.cn/jsref/jsref_escape.asp) | 对字符串进行编码。 |
| [eval()](http://www.w3school.com.cn/jsref/jsref_eval.asp) | 计算 JavaScript 字符串，并把它作为脚本代码来执行。 |
| [getClass()](http://www.w3school.com.cn/jsref/jsref_getClass.asp) | 返回一个 JavaObject 的 JavaClass。 |
| [isFinite()](http://www.w3school.com.cn/jsref/jsref_isFinite.asp) | 检查某个值是否为有穷大的数。 |
| [isNaN()](http://www.w3school.com.cn/jsref/jsref_isNaN.asp) | 检查某个值是否是数字。 |
| [Number()](http://www.w3school.com.cn/jsref/jsref_number.asp) | 把对象的值转换为数字。 |
| [parseFloat()](http://www.w3school.com.cn/jsref/jsref_parseFloat.asp) | 解析一个字符串并返回一个浮点数。 |
| [parseInt()](http://www.w3school.com.cn/jsref/jsref_parseInt.asp) | 解析一个字符串并返回一个整数。 |
| [String()](http://www.w3school.com.cn/jsref/jsref_string.asp) | 把对象的值转换为字符串。 |
| [unescape()](http://www.w3school.com.cn/jsref/jsref_unescape.asp) | 对由 escape() 编码的字符串进行解码。 |

#### 顶层属性（全局属性）

|  |  |
| --- | --- |
| **方法** | **描述** |
| [Infinity](http://www.w3school.com.cn/jsref/jsref_infinity.asp) | 代表正的无穷大的数值。 |
| [java](http://www.w3school.com.cn/jsref/jsref_java.asp) | 代表 java.\* 包层级的一个 JavaPackage。 |
| [NaN](http://www.w3school.com.cn/jsref/jsref_nan.asp) | 指示某个值是不是数字值。 |
| [Packages](http://www.w3school.com.cn/jsref/jsref_Packages.asp) | 根 JavaPackage 对象。 |
| [undefined](http://www.w3school.com.cn/jsref/jsref_undefined.asp) | 指示未定义的值。 |

#### 全局对象描述

全局对象是预定义的对象，作为 JavaScript 的全局函数和全局属性的占位符。通过使用全局对象，可以访问所有其他所有预定义的对象、函数和属性。全局对象不是任何对象的属性，所以它没有名称。

在顶层 JavaScript 代码中，可以用关键字 this 引用全局对象。但通常不必用这种方式引用全局对象，因为全局对象是作用域链的头，这意味着所有非限定性的变量和函数名都会作为该对象的属性来查询。例如，当JavaScript 代码引用 parseInt() 函数时，它引用的是全局对象的 parseInt 属性。全局对象是作用域链的头，还意味着在顶层 JavaScript 代码中声明的所有变量都将成为全局对象的属性。

全局对象只是一个对象，而不是类。既没有构造函数，也无法实例化一个新的全局对象。

在 JavaScript 代码嵌入一个特殊环境中时，全局对象通常具有环境特定的属性。实际上，ECMAScript 标准没有规定全局对象的类型，JavaScript 的实现或嵌入的 JavaScript 都可以把任意类型的对象作为全局对象，只要该对象定义了这里列出的基本属性和函数。例如，在允许通过 LiveConnect 或相关的技术来脚本化 Java 的 JavaScript 实现中，全局对象被赋予了这里列出的 java 和 Package 属性以及 getClass() 方法。而在客户端 JavaScript 中，全局对象就是 Window 对象，表示允许 JavaScript 代码的 Web 浏览器窗口。

**例子**

在 JavaScript 核心语言中，全局对象的预定义属性都是不可枚举的，所有可以用 for/in 循环列出所有隐式或显式声明的全局变量，如下所示：

var variables = "";

for (var name in this)

{

variables += name + "<br />";

}

document.write(variables);

### 3.10 JavaScript 事件参考手册

**事件通常与函数配合使用，这样就可以通过发生的事件来驱动函数执行。**

#### 事件句柄

HTML 4.0 的新特性之一是有能力使 HTML 事件触发浏览器中的动作（action），比如当用户点击某个 HTML 元素时启动一段 JavaScript。下面是一个属性列表，这些属性可插入 HTML 标签来定义事件动作。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性** | **当以下情况发生时，出现此事件** | **FF** | **N** | **IE** |
| onabort | 图像加载被中断 | 1 | 3 | 4 |
| onblur | 元素失去焦点 | 1 | 2 | 3 |
| onchange | 用户改变域的内容 | 1 | 2 | 3 |
| onclick | 鼠标点击某个对象 | 1 | 2 | 3 |
| ondblclick | 鼠标双击某个对象 | 1 | 4 | 4 |
| onerror | 当加载文档或图像时发生某个错误 | 1 | 3 | 4 |
| onfocus | 元素获得焦点 | 1 | 2 | 3 |
| onkeydown | 某个键盘的键被按下 | 1 | 4 | 3 |
| onkeypress | 某个键盘的键被按下或按住 | 1 | 4 | 3 |
| onkeyup | 某个键盘的键被松开 | 1 | 4 | 3 |
| onload | 某个页面或图像被完成加载 | 1 | 2 | 3 |
| onmousedown | 某个鼠标按键被按下 | 1 | 4 | 4 |
| onmousemove | 鼠标被移动 | 1 | 6 | 3 |
| onmouseout | 鼠标从某元素移开 | 1 | 4 | 4 |
| onmouseover | 鼠标被移到某元素之上 | 1 | 2 | 3 |
| onmouseup | 某个鼠标按键被松开 | 1 | 4 | 4 |
| onreset | 重置按钮被点击 | 1 | 3 | 4 |
| onresize | 窗口或框架被调整尺寸 | 1 | 4 | 4 |
| onselect | 文本被选定 | 1 | 2 | 3 |
| onsubmit | 提交按钮被点击 | 1 | 2 | 3 |
| onunload | 用户退出页面 | 1 | 2 | 3 |

四、JavaScript Window

### 4.1 JavaScript Window - 浏览器对象模型

**浏览器对象模型 (BOM) 使 JavaScript 有能力与浏览器“对话”。**

**浏览器对象模型 (BOM)**

浏览器对象模型（Browser Object Model）尚无正式标准。

由于现代浏览器已经（几乎）实现了 JavaScript 交互性方面的相同方法和属性，因此常被认为是 BOM 的方法和属性。

#### Window 对象

所有浏览器都支持 **window** 对象。它表示浏览器窗口。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。

全局变量是 window 对象的属性。

全局函数是 window 对象的方法。

甚至 HTML DOM 的 document 也是 window 对象的属性之一：

window.document.getElementById("header");

与此相同：

document.getElementById("header");

#### Window 尺寸

有三种方法能够确定浏览器窗口的尺寸（浏览器的视口，不包括工具栏和滚动条）。

对于Internet Explorer、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

对于 Internet Explorer 8、7、6、5：

* document.documentElement.clientHeight
* document.documentElement.clientWidth

或者

* document.body.clientHeight
* document.body.clientWidth

实用的 JavaScript 方案（涵盖所有浏览器）：

**实例**

var w=window.innerWidth

|| document.documentElement.clientWidth

|| document.body.clientWidth;

var h=window.innerHeight

|| document.documentElement.clientHeight

|| document.body.clientHeight;

#### Window 对象集合

|  |  |
| --- | --- |
| **集合** | **描述** |
| frames[] | 返回窗口中所有命名的框架。  该集合是 Window 对象的数组，每个 Window 对象在窗口中含有一个框架或 <iframe>。属性 frames.length 存放数组 frames[] 中含有的元素个数。注意，frames[] 数组中引用的框架可能还包括框架，它们自己也具有 frames[] 数组。 |

#### Window 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [closed](http://www.w3school.com.cn/jsref/prop_win_closed.asp) | 返回窗口是否已被关闭。 |
| [defaultStatus](http://www.w3school.com.cn/jsref/prop_win_defaultstatus.asp) | 设置或返回窗口状态栏中的默认文本。 |
| [document](http://www.w3school.com.cn/jsref/dom_obj_document.asp) | 对 Document 对象的只读引用。请参阅 [Document 对象](http://www.w3school.com.cn/jsref/dom_obj_document.asp)。 |
| [history](http://www.w3school.com.cn/jsref/dom_obj_history.asp) | 对 History 对象的只读引用。请参数 [History 对象](http://www.w3school.com.cn/jsref/dom_obj_history.asp)。 |
| [innerheight](http://www.w3school.com.cn/jsref/prop_win_innerheight_innerwidth.asp) | 返回窗口的文档显示区的高度。 |
| [innerwidth](http://www.w3school.com.cn/jsref/prop_win_innerheight_innerwidth.asp) | 返回窗口的文档显示区的宽度。 |
| length | 设置或返回窗口中的框架数量。 |
| [location](http://www.w3school.com.cn/jsref/dom_obj_location.asp) | 用于窗口或框架的 Location 对象。请参阅 [Location 对象](http://www.w3school.com.cn/jsref/dom_obj_location.asp)。 |
| [name](http://www.w3school.com.cn/jsref/prop_win_name.asp) | 设置或返回窗口的名称。 |
| [Navigator](http://www.w3school.com.cn/jsref/dom_obj_navigator.asp) | 对 Navigator 对象的只读引用。请参数 [Navigator 对象](http://www.w3school.com.cn/jsref/dom_obj_navigator.asp)。 |
| [opener](http://www.w3school.com.cn/jsref/prop_win_opener.asp) | 返回对创建此窗口的窗口的引用。 |
| [outerheight](http://www.w3school.com.cn/jsref/prop_win_outerheight.asp) | 返回窗口的外部高度。 |
| [outerwidth](http://www.w3school.com.cn/jsref/prop_win_outerwidth.asp) | 返回窗口的外部宽度。 |
| pageXOffset | 设置或返回当前页面相对于窗口显示区左上角的 X 位置。 |
| pageYOffset | 设置或返回当前页面相对于窗口显示区左上角的 Y 位置。 |
| parent | 返回父窗口。 |
| [Screen](http://www.w3school.com.cn/jsref/dom_obj_screen.asp) | 对 Screen 对象的只读引用。请参数 [Screen 对象](http://www.w3school.com.cn/jsref/dom_obj_screen.asp)。 |
| [self](http://www.w3school.com.cn/jsref/prop_win_self.asp) | 返回对当前窗口的引用。等价于 Window 属性。 |
| [status](http://www.w3school.com.cn/jsref/prop_win_status.asp) | 设置窗口状态栏的文本。 |
| [top](http://www.w3school.com.cn/jsref/prop_win_top.asp) | 返回最顶层的先辈窗口。 |
| window | window 属性等价于 self 属性，它包含了对窗口自身的引用。 |
| * screenLeft * screenTop * screenX * screenY | 只读整数。声明了窗口的左上角在屏幕上的的 x 坐标和 y 坐标。IE、Safari 和 Opera 支持 screenLeft 和 screenTop，而 Firefox 和 Safari 支持 screenX 和 screenY。 |

#### Window 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [alert()](http://www.w3school.com.cn/jsref/met_win_alert.asp) | 显示带有一段消息和一个确认按钮的警告框。 |
| [blur()](http://www.w3school.com.cn/jsref/met_win_blur.asp) | 把键盘焦点从顶层窗口移开。 |
| [clearInterval()](http://www.w3school.com.cn/jsref/met_win_clearinterval.asp) | 取消由 setInterval() 设置的 timeout。 |
| [clearTimeout()](http://www.w3school.com.cn/jsref/met_win_cleartimeout.asp) | 取消由 setTimeout() 方法设置的 timeout。 |
| [close()](http://www.w3school.com.cn/jsref/met_win_close.asp) | 关闭浏览器窗口。 |
| [confirm()](http://www.w3school.com.cn/jsref/met_win_confirm.asp) | 显示带有一段消息以及确认按钮和取消按钮的对话框。 |
| [createPopup()](http://www.w3school.com.cn/jsref/met_win_createpopup.asp) | 创建一个 pop-up 窗口。 |
| [focus()](http://www.w3school.com.cn/jsref/met_win_focus.asp) | 把键盘焦点给予一个窗口。 |
| [moveBy()](http://www.w3school.com.cn/jsref/met_win_moveby.asp) | 可相对窗口的当前坐标把它移动指定的像素。 |
| [moveTo()](http://www.w3school.com.cn/jsref/met_win_moveto.asp) | 把窗口的左上角移动到一个指定的坐标。 |
| [open()](http://www.w3school.com.cn/jsref/met_win_open.asp) | 打开一个新的浏览器窗口或查找一个已命名的窗口。 |
| [print()](http://www.w3school.com.cn/jsref/met_win_print.asp) | 打印当前窗口的内容。 |
| [prompt()](http://www.w3school.com.cn/jsref/met_win_prompt.asp) | 显示可提示用户输入的对话框。 |
| [resizeBy()](http://www.w3school.com.cn/jsref/met_win_resizeby.asp) | 按照指定的像素调整窗口的大小。 |
| [resizeTo()](http://www.w3school.com.cn/jsref/met_win_resizeto.asp) | 把窗口的大小调整到指定的宽度和高度。 |
| [scrollBy()](http://www.w3school.com.cn/jsref/met_win_scrollby.asp) | 按照指定的像素值来滚动内容。 |
| [scrollTo()](http://www.w3school.com.cn/jsref/met_win_scrollto.asp) | 把内容滚动到指定的坐标。 |
| [setInterval()](http://www.w3school.com.cn/jsref/met_win_setinterval.asp) | 按照指定的周期（以毫秒计）来调用函数或计算表达式。 |
| [setTimeout()](http://www.w3school.com.cn/jsref/met_win_settimeout.asp) | 在指定的毫秒数后调用函数或计算表达式。 |

#### Window 对象描述

Window 对象表示一个浏览器窗口或一个框架。在客户端 JavaScript 中，Window 对象是全局对象，所有的表达式都在当前的环境中计算。也就是说，要引用当前窗口根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来使用。例如，可以只写[document](http://www.w3school.com.cn/jsref/dom_obj_document.asp)，而不必写 window.document。

同样，可以把当前窗口对象的方法当作函数来使用，如只写 alert()，而不必写 Window.alert()。

除了上面列出的属性和方法，Window 对象还实现了核心 JavaScript 所定义的所有全局属性和方法。

Window 对象的 window 属性和 [self 属性](http://www.w3school.com.cn/jsref/prop_win_self.asp)引用的都是它自己。当你想明确地引用当前窗口，而不仅仅是隐式地引用它时，可以使用这两个属性。除了这两个属性之外，parent 属性、top 属性以及 frame[] 数组都引用了与当前 Window 对象相关的其他 Window 对象。

要引用窗口中的一个框架，可以使用如下语法：

frame[i] //当前窗口的框架

self.frame[i] //当前窗口的框架

w.frame[i] //窗口 w 的框架

要引用一个框架的父窗口（或父框架），可以使用下面的语法：

parent //当前窗口的父窗口

self.parent //当前窗口的父窗口

w.parent //窗口 w 的父窗口

要从顶层窗口含有的任何一个框架中引用它，可以使用如下语法：

top //当前框架的顶层窗口

self.top //当前框架的顶层窗口

f.top //框架 f 的顶层窗口

新的顶层浏览器窗口由方法 Window.open() 创建。当调用该方法时，应把 open() 调用的返回值存储在一个变量中，然后使用那个变量来引用新窗口。新窗口的 [opener 属性](http://www.w3school.com.cn/jsref/prop_win_opener.asp)反过来引用了打开它的那个窗口。

一般来说，Window 对象的方法都是对浏览器窗口或框架进行某种操作。而 [alert() 方法](http://www.w3school.com.cn/jsref/met_win_alert.asp)、[confirm() 方法](http://www.w3school.com.cn/jsref/met_win_confirm.asp)和 [prompt 方法](http://www.w3school.com.cn/jsref/met_win_prompt.asp)则不同，它们通过简单的对话框与用户进行交互。

### 4.2 JavaScript Window Screen

**window.screen 对象包含有关用户屏幕的信息。**

一些属性：

* screen.availWidth - 可用的屏幕宽度
* screen.availHeight - 可用的屏幕高度

#### Screen 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [availHeight](http://www.w3school.com.cn/jsref/prop_screen_availheight.asp) | 返回显示屏幕的高度 (除 Windows 任务栏之外)。 |
| [availWidth](http://www.w3school.com.cn/jsref/prop_screen_availwidth.asp) | 返回显示屏幕的宽度 (除 Windows 任务栏之外)。 |
| [bufferDepth](http://www.w3school.com.cn/jsref/prop_screen_bufferdepth.asp) | 设置或返回调色板的比特深度。 |
| [colorDepth](http://www.w3school.com.cn/jsref/prop_screen_colordepth.asp) | 返回目标设备或缓冲器上的调色板的比特深度。 |
| [deviceXDPI](http://www.w3school.com.cn/jsref/prop_screen_devicexdpi.asp) | 返回显示屏幕的每英寸水平点数。 |
| [deviceYDPI](http://www.w3school.com.cn/jsref/prop_screen_deviceydpi.asp) | 返回显示屏幕的每英寸垂直点数。 |
| [fontSmoothingEnabled](http://www.w3school.com.cn/jsref/prop_screen_fontsmoothingenabled.asp) | 返回用户是否在显示控制面板中启用了字体平滑。 |
| [height](http://www.w3school.com.cn/jsref/prop_screen_height.asp) | 返回显示屏幕的高度。 |
| [logicalXDPI](http://www.w3school.com.cn/jsref/prop_screen_logicalxdpi.asp) | 返回显示屏幕每英寸的水平方向的常规点数。 |
| [logicalYDPI](http://www.w3school.com.cn/jsref/prop_screen_logicalydpi.asp) | 返回显示屏幕每英寸的垂直方向的常规点数。 |
| [pixelDepth](http://www.w3school.com.cn/jsref/prop_screen_pixeldepth.asp) | 返回显示屏幕的颜色分辨率（比特每像素）。 |
| [updateInterval](http://www.w3school.com.cn/jsref/prop_screen_updateinterval.asp) | 设置或返回屏幕的刷新率。 |
| [width](http://www.w3school.com.cn/jsref/prop_screen_width.asp) | 返回显示器屏幕的宽度。 |

#### Screen 对象描述

每个 Window 对象的 screen 属性都引用一个 Screen 对象。Screen 对象中存放着有关显示浏览器屏幕的信息。JavaScript 程序将利用这些信息来优化它们的输出，以达到用户的显示要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，它还可以根据显示器的颜色深度选择使用 16 位色还是使用 8 位色的图形。另外，JavaScript 程序还能根据有关屏幕尺寸的信息将新的浏览器窗口定位在屏幕中间。

### 4.3 JavaScript Window Location

window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

**window.location** 对象在编写时可不使用 window 这个前缀。

一些例子：

* location.hostname 返回 web 主机的域名
* location.pathname 返回当前页面的路径和文件名
* location.port 返回 web 主机的端口 （80 或 443）
* location.protocol 返回所使用的 web 协议（http:// 或 https://）
* location.href 属性返回当前页面的 URL。
* location.assign() 方法加载新的文档。

加载一个新的文档：

<html>

<head>

<script>

function newDoc()

{

window.location.assign("http://www.w3school.com.cn")

}

</script>

</head>

<body>

<input type="button" value="加载新文档" onclick="newDoc()">

</body>

</html>

#### Location 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [hash](http://www.w3school.com.cn/jsref/prop_loc_hash.asp) | 设置或返回从井号 (#) 开始的 URL（锚）。 |
| [host](http://www.w3school.com.cn/jsref/prop_loc_host.asp) | 设置或返回主机名和当前 URL 的端口号。 |
| [hostname](http://www.w3school.com.cn/jsref/prop_loc_hostname.asp) | 设置或返回当前 URL 的主机名。 |
| [href](http://www.w3school.com.cn/jsref/prop_loc_href.asp) | 设置或返回完整的 URL。 |
| [pathname](http://www.w3school.com.cn/jsref/prop_loc_pathname.asp) | 设置或返回当前 URL 的路径部分。 |
| [port](http://www.w3school.com.cn/jsref/prop_loc_port.asp) | 设置或返回当前 URL 的端口号。 |
| [protocol](http://www.w3school.com.cn/jsref/prop_loc_protocol.asp) | 设置或返回当前 URL 的协议。 |
| [search](http://www.w3school.com.cn/jsref/prop_loc_search.asp) | 设置或返回从问号 (?) 开始的 URL（查询部分）。 |

#### Location 对象方法

|  |  |
| --- | --- |
| **属性** | **描述** |
| [assign()](http://www.w3school.com.cn/jsref/met_loc_assign.asp) | 加载新的文档。 |
| [reload()](http://www.w3school.com.cn/jsref/met_loc_reload.asp) | 重新加载当前文档。 |
| [replace()](http://www.w3school.com.cn/jsref/met_loc_replace.asp) | 用新的文档替换当前文档。 |

#### Location 对象描述

Location 对象存储在 Window 对象的 Location 属性中，表示那个窗口中当前显示的文档的 Web 地址。它的 [href 属性](http://www.w3school.com.cn/jsref/prop_loc_href.asp)存放的是文档的完整 URL，其他属性则分别描述了 URL 的各个部分。这些属性与 Anchor 对象（或 Area 对象）的 URL 属性非常相似。当一个 Location 对象被转换成字符串，href 属性的值被返回。这意味着你可以使用表达式 location 来替代 location.href。

不过 Anchor 对象表示的是文档中的超链接，Location 对象表示的却是浏览器当前显示的文档的 URL（或位置）。但是 Location 对象所能做的远远不止这些，它还能控制浏览器显示的文档的位置。如果把一个含有 URL 的字符串赋予 Location 对象或它的 href 属性，浏览器就会把新的 URL 所指的文档装载进来，并显示出来。

除了设置 location 或 location.href 用完整的 URL 替换当前的 URL 之外，还可以修改部分 URL，只需要给 Location 对象的其他属性赋值即可。这样做就会创建新的 URL，其中的一部分与原来的 URL 不同，浏览器会将它装载并显示出来。例如，假设设置了Location对象的 [hash 属性](http://www.w3school.com.cn/jsref/prop_loc_hash.asp)，那么浏览器就会转移到当前文档中的一个指定的位置。同样，如果设置了 [search 属性](http://www.w3school.com.cn/jsref/prop_loc_search.asp)，那么浏览器就会重新装载附加了新的查询字符串的 URL。

除了 URL 属性外，Location 对象的 [reload() 方法](http://www.w3school.com.cn/jsref/met_loc_reload.asp)可以重新装载当前文档，[replace()](http://www.w3school.com.cn/jsref/met_loc_replace.asp) 可以装载一个新文档而无须为它创建一个新的历史记录，也就是说，在浏览器的历史列表中，新文档将替换当前文档。

### 4.4 JavaScript Window History

window.history 对象包含浏览器的历史。

**window.history** 对象在编写时可不使用 window 这个前缀。

一些方法：

* history.back() - 与在浏览器点击后退按钮相同
* history.forward() - 与在浏览器中点击按钮向前相同

#### History 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [length](http://www.w3school.com.cn/jsref/prop_his_length.asp) | 返回浏览器历史列表中的 URL 数量。 |

#### History 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [back()](http://www.w3school.com.cn/jsref/met_his_back.asp) | 加载 history 列表中的前一个 URL。 |
| [forward()](http://www.w3school.com.cn/jsref/met_his_forward.asp) | 加载 history 列表中的下一个 URL。 |
| [go()](http://www.w3school.com.cn/jsref/met_his_go.asp) | 加载 history 列表中的某个具体页面。 |

#### History 对象描述

History 对象最初设计来表示窗口的浏览历史。但出于隐私方面的原因，History 对象不再允许脚本访问已经访问过的实际 URL。唯一保持使用的功能只有 [back()](http://www.w3school.com.cn/jsref/met_his_back.asp)、[forward()](http://www.w3school.com.cn/jsref/met_his_forward.asp) 和 [go()](http://www.w3school.com.cn/jsref/met_his_go.asp) 方法。

**例子**

下面一行代码执行的操作与单击后退按钮执行的操作一样：

history.back()

下面一行代码执行的操作与单击两次后退按钮执行的操作一样：

history.go(-2)

### 4.5 JavaScript Window Navigator

window.navigator 对象包含有关访问者浏览器的信息。

<div id="example"></div>

<script>

txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";

txt+= "<p>Browser Name: " + navigator.appName + "</p>";

txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";

txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";

txt+= "<p>Platform: " + navigator.platform + "</p>";

txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";

txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>";

document.getElementById("example").innerHTML=txt;

</script>

**警告：**来自 navigator 对象的信息具有误导性，不应该被用于检测浏览器版本，这是因为：

* navigator 数据可被浏览器使用者更改
* 浏览器无法报告晚于浏览器发布的新操作系统

#### Navigator 对象集合

|  |  |
| --- | --- |
| **集合** | **描述** |
| plugins[] | 返回对文档中所有嵌入式对象的引用。  该集合是一个 Plugin 对象的数组，其中的元素代表浏览器已经安装的插件。Plug-in 对象提供的是有关插件的信息，其中包括它所支持的 MIME 类型的列表。  虽然 plugins[] 数组是由 IE 4 定义的，但是在 IE 4 中它却总是空的，因为 IE 4 不支持插件和 Plugin 对象。 |

#### Navigator 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [appCodeName](http://www.w3school.com.cn/jsref/prop_nav_appcodename.asp) | 返回浏览器的代码名。 |
| [appMinorVersion](http://www.w3school.com.cn/jsref/prop_nav_appminorversion.asp) | 返回浏览器的次级版本。 |
| [appName](http://www.w3school.com.cn/jsref/prop_nav_appname.asp) | 返回浏览器的名称。 |
| [appVersion](http://www.w3school.com.cn/jsref/prop_nav_appversion.asp) | 返回浏览器的平台和版本信息。 |
| [browserLanguage](http://www.w3school.com.cn/jsref/prop_nav_browserlanguage.asp) | 返回当前浏览器的语言。 |
| [cookieEnabled](http://www.w3school.com.cn/jsref/prop_nav_cookieenabled.asp) | 返回指明浏览器中是否启用 cookie 的布尔值。 |
| [cpuClass](http://www.w3school.com.cn/jsref/prop_nav_cpuclass.asp) | 返回浏览器系统的 CPU 等级。 |
| [onLine](http://www.w3school.com.cn/jsref/prop_nav_online.asp) | 返回指明系统是否处于脱机模式的布尔值。 |
| [platform](http://www.w3school.com.cn/jsref/prop_nav_platform.asp) | 返回运行浏览器的操作系统平台。 |
| [systemLanguage](http://www.w3school.com.cn/jsref/prop_nav_systemlanguage.asp) | 返回 OS 使用的默认语言。 |
| [userAgent](http://www.w3school.com.cn/jsref/prop_nav_useragent.asp) | 返回由客户机发送服务器的 user-agent 头部的值。 |
| [userLanguage](http://www.w3school.com.cn/jsref/prop_nav_userlanguage.asp) | 返回 OS 的自然语言设置。 |

#### Navigator 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [javaEnabled()](http://www.w3school.com.cn/jsref/met_nav_javaenabled.asp) | 规定浏览器是否启用 Java。 |
| [taintEnabled()](http://www.w3school.com.cn/jsref/met_nav_taintenabled.asp) | 规定浏览器是否启用数据污点 (data tainting)。 |

#### Navigator 对象描述

Navigator 对象包含的属性描述了正在使用的浏览器。可以使用这些属性进行平台专用的配置。

虽然这个对象的名称显而易见的是 Netscape 的 Navigator 浏览器，但其他实现了 JavaScript 的浏览器也支持这个对象。

Navigator 对象的实例是唯一的，可以用 Window 对象的 navigator 属性来引用它。

### 4.6 JavaScript 消息框

**可以在 JavaScript 中创建三种消息框：警告框、确认框、提示框。**

#### 警告框

警告框经常用于确保用户可以得到某些信息。

当警告框出现后，用户需要点击确定按钮才能继续进行操作。

**语法：**

alert("文本")

#### 确认框

确认框用于使用户可以验证或者接受某些信息。

当确认框出现后，用户需要点击确定或者取消按钮才能继续进行操作。

如果用户点击确认，那么返回值为 true。如果用户点击取消，那么返回值为 false。

**语法：**

confirm("文本")

#### 提示框

提示框经常用于提示用户在进入页面前输入某个值。

当提示框出现后，用户需要输入某个值，然后点击确认或取消按钮才能继续操纵。

如果用户点击确认，那么返回值为输入的值。如果用户点击取消，那么返回值为 null。

**语法：**

prompt("文本","默认值")

### 4.7 JavaScript 计时

**通过使用 JavaScript，我们有能力做到在一个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。我们称之为计时事件。**

#### JavaScript 计时事件

通过使用 JavaScript，我们有能力作到在一个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。我们称之为计时事件。

在 JavaScritp 中使用计时事件是很容易的，两个关键方法是:

**setTimeout()**

未来的某时执行代码

**clearTimeout()**

取消setTimeout()

#### setTimeout()

**语法**

var t=setTimeout("javascript语句",毫秒)

setTimeout() 方法会返回某个值。在上面的语句中，值被储存在名为 t 的变量中。假如你希望取消这个 setTimeout()，你可以使用这个变量名来指定它。

setTimeout() 的第一个参数是含有 JavaScript 语句的字符串。这个语句可能诸如 "alert('5 seconds!')"，或者对函数的调用，诸如 alertMsg()"。

第二个参数指示从当前起多少毫秒后执行第一个参数。

**提示：**1000 毫秒等于一秒。

#### clearTimeout()

**语法**

clearTimeout(setTimeout\_variable)

### 4.8 JavaScript Cookies

**cookie 用来识别用户。**

cookie 是存储于访问者的计算机中的变量。每当同一台计算机通过浏览器请求某个页面时，就会发送这个 cookie。你可以使用 JavaScript 来创建和取回 cookie 的值。

#### 创建和存储 cookie

在这个例子中我们要创建一个存储访问者名字的 cookie。当访问者首次访问网站时，他们会被要求填写姓名。名字会存储于 cookie 中。当访问者再次访问网站时，他们就会收到欢迎词。

首先，我们会创建一个可在 cookie 变量中存储访问者姓名的函数：

function setCookie(c\_name,value,expiredays)

{

var exdate=new Date()

exdate.setDate(exdate.getDate()+expiredays)

document.cookie=c\_name+ "=" +escape(value)+

((expiredays==null) ? "" : ";expires="+exdate.toGMTString())

}

上面这个函数中的参数存有 cookie 的名称、值以及过期天数。

在上面的函数中，我们首先将天数转换为有效的日期，然后，我们将 cookie 名称、值及其过期日期存入 document.cookie 对象。

之后，我们要创建另一个函数来检查是否已设置 cookie：

function getCookie(c\_name)

{

if (document.cookie.length>0)

{

c\_start=document.cookie.indexOf(c\_name + "=")

if (c\_start!=-1)

{

c\_start=c\_start + c\_name.length+1

c\_end=document.cookie.indexOf(";",c\_start)

if (c\_end==-1) c\_end=document.cookie.length

return unescape(document.cookie.substring(c\_start,c\_end))

}

}

return ""

}

上面的函数首先会检查 document.cookie 对象中是否存有 cookie。假如 document.cookie 对象存有某些 cookie，那么会继续检查我们指定的 cookie 是否已储存。如果找到了我们要的 cookie，就返回值，否则返回空字符串。

最后，我们要创建一个函数，这个函数的作用是：如果 cookie 已设置，则显示欢迎词，否则显示提示框来要求用户输入名字。

function checkCookie()

{

username=getCookie('username')

if (username!=null && username!="")

{alert('Welcome again '+username+'!')}

else

{

username=prompt('Please enter your name:',"")

if (username!=null && username!="")

{

setCookie('username',username,365)

}

}

}

这是所有的代码：

<html>

<head>

<script type="text/javascript">

function getCookie(c\_name)

{

if (document.cookie.length>0)

{

c\_start=document.cookie.indexOf(c\_name + "=")

if (c\_start!=-1)

{

c\_start=c\_start + c\_name.length+1

c\_end=document.cookie.indexOf(";",c\_start)

if (c\_end==-1) c\_end=document.cookie.length

return unescape(document.cookie.substring(c\_start,c\_end))

}

}

return ""

}

function setCookie(c\_name,value,expiredays)

{

var exdate=new Date()

exdate.setDate(exdate.getDate()+expiredays)

document.cookie=c\_name+ "=" +escape(value)+

((expiredays==null) ? "" : ";expires="+exdate.toGMTString())

}

function checkCookie()

{

username=getCookie('username')

if (username!=null && username!="")

{alert('Welcome again '+username+'!')}

else

{

username=prompt('Please enter your name:',"")

if (username!=null && username!="")

{

setCookie('username',username,365)

}

}

}

</script>

</head>

<body onLoad="checkCookie()">

</body>

</html>

五、JavaScript 库

### 5.1 JavaScript 库

JavaScript 库 - jQuery、Prototype、MooTools。

#### JavaScript 框架（库）

JavaScript 高级程序设计（特别是对浏览器差异的复杂处理），通常很困难也很耗时。

为了应对这些调整，许多的 **JavaScript (helper) 库**应运而生。

这些 JavaScript 库常被称为 **JavaScript 框架**。

在本教程中，我们将了解到一些广受欢迎的 JavaScript 框架：

* jQuery
* Prototype
* MooTools

所有这些框架都提供针对常见 JavaScript 任务的函数，包括动画、DOM 操作以及 Ajax 处理。

#### jQuery

jQuery 是目前最受欢迎的 JavaScript 框架。

它使用 CSS 选择器来访问和操作网页上的 HTML 元素（DOM 对象）。

jQuery 同时提供 companion UI（用户界面）和插件。

许多大公司在网站上使用 jQuery：

* Google
* Microsoft
* IBM
* Netflix

#### Prototype

**Prototype** 是一种库，提供用于执行常见 web 任务的简单 API。

**API** 是应用程序编程接口（Application Programming Interface）的缩写。它是包含属性和方法的库，用于操作 HTML DOM。

Prototype 通过提供类和继承，实现了对 JavaScript 的增强。

#### MooTools

**MooTools** 也是一个框架，提供了可使常见的 JavaScript 编程更为简单的 API。

MooTools 也含有一些轻量级的效果和动画函数。

#### 其他框架

下面是其他一些在上面未涉及的框架：

**YUI** - Yahoo! User Interface Framework，涵盖大量函数的大型库，从简单的 JavaScript 功能到完整的 internet widget。

**Ext JS** - 可定制的 widget，用于构建富因特网应用程序（rich Internet applications）。

**Dojo** - 用于 DOM 操作、事件、widget 等的工具包。

**script.aculo.us** - 开源的 JavaScript 框架，针对可视效果和界面行为。

**UIZE** - Widget、AJAX、DOM、模板等等。

#### CDN - 内容分发网络

您总是希望网页可以尽可能地快。您希望页面的容量尽可能地小，同时您希望浏览器尽可能多地进行缓存。

如果许多不同的网站使用相同的 JavaScript 框架，那么把框架库存放在一个通用的位置供每个网页分享就变得很有意义了。

CDN (Content Delivery Network) 解决了这个问题。CDN 是包含可分享代码库的服务器网络。

Google 为一系列 JavaScript 库提供了免费的 CDN，包括：

* jQuery
* Prototype
* MooTools
* Dojo
* Yahoo! YUI

如需在您的网页中使用 JavaScript 框架库，只需在 <script> 标签中引用该库即可：

#### 引用 jQuery

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js">

</script>

### 5.2 JavaScript - 测试 jQuery

#### 引用 jQuery

如需测试 JavaScript 库，您需要在网页中引用它。

为了引用某个库，请使用 <script> 标签，其 src 属性设置为库的 URL：

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js">

</script>

</head>

<body>

</body>

</html>

#### jQuery 描述

主要的 jQuery 函数是 $() 函数（jQuery 函数）。如果您向该函数传递 DOM 对象，它会返回 jQuery 对象，带有向其添加的 jQuery 功能。

jQuery 允许您通过 CSS 选择器来选取元素。

在 JavaScript 中，您可以分配一个函数以处理窗口加载事件：

##### JavaScript 方式：

function myFunction()

{

var obj=document.getElementById("h01");

obj.innerHTML="Hello jQuery";

}

onload=myFunction;

等价的 jQuery 是不同的：

##### jQuery 方式：

function myFunction()

{

$("#h01").html("Hello jQuery");

}

$(document).ready(myFunction);

上面代码的最后一行，HTML DOM 文档对象被传递到 jQuery ：$(document)。

当您向 jQuery 传递 DOM 对象时，jQuery 会返回以 HTML DOM 对象包装的 jQuery 对象。

jQuery 函数会返回新的 jQuery 对象，其中的 ready() 是一个方法。

由于在 JavaScript 中函数就是变量，因此可以把 myFunction 作为变量传递给 jQuery 的 ready 方法。

**提示：**jQuery 返回 jQuery 对象，与已传递的 DOM 对象不同。jQuery 对象拥有的属性和方法，与 DOM 对象的不同。您不能在 jQuery 对象上使用 HTML DOM 的属性和方法。

### 5.3 JavaScript - 测试 Prototype

#### 引用 Prototype

如需测试 JavaScript 库，您需要在网页中引用它。

为了引用某个库，请使用 <script> 标签，其 src 属性设置为库的 URL：

<!DOCTYPE html>

<html>

<head>

<script src="http://ajax.googleapis.com/ajax/libs/prototype/1.7.1.0/prototype.js>

</script>

</head>

<body>

</body>

</html>

#### Prototype 描述

Prototype 提供的函数可使 HTML DOM 编程更容易。

与 jQuery 类似，Prototype 也有自己的 $() 函数。

$() 函数接受 HTML DOM 元素的 id 值（或 DOM 元素），并会向 DOM 对象添加新的功能。

与 jQuery 不同，Prototype 没有用以取代 window.onload() 的 ready() 方法。相反，Prototype 会向浏览器及 HTML DOM 添加扩展。

在 JavaScript 中，您可以分配一个函数以处理窗口加载事件：

##### JavaScript 方式：

function myFunction()

{

var obj=document.getElementById("h01");

obj.innerHTML="Hello Prototype";

}

onload=myFunction;

等价的 Prototype 是不同的：

##### Prototype 方式：

function myFunction()

{

$("h01").insert("Hello Prototype!");

}

Event.observe(window,"load",myFunction);

Event.observe() 接受三个参数：

* 您希望处理的 HTML DOM 或 BOM（浏览器对象模型）对象
* 您希望处理的事件
* 您希望调用的函数