# JAVASCRIPT

- **JAVASCRIPT 是世界上最流行的编程语言。**
- **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。
- JAVASCRIPT 很容易学习。

# 写入 HTML 输出(1写入HTML输出.HTML)

```
<!DOCTYPE html>
<html>
<body>

<p>
JavaScript 能够直接写入 HTML 输出流中：
</p>

<script>
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
</script>

<p>
您只能在 HTML 输出流中使用 <strong>document.write</strong>。
如果您在文档已加载后使用它（比如在函数中），会覆盖整个文档。
</p>

</body>
</html>
```

# 对事件作出反应(2对事件作出反应.HTML)

```
<!DOCTYPE html>
<html>
<body>

<h1>我的第一段 JavaScript</h1>

<p>
JavaScript 能够对事件作出反应。比如对按钮的点击:
</p>

<button type="button" onclick="alert('Welcome!')">点击这里</button>

</body>
</html>
```

# 改变 HTML内容(3改变 HTML内容.HTML)

```
<!DOCTYPE html>
<html>
<body>

<h1>我的第一段 JavaScript</h1>

<p id="demo">
JavaScript 能改变 HTML 元素的内容。
</p>

<script>
function myFunction()
{
x=document.getElementById("demo");   // 找到元素
x.innerHTML="Hello JavaScript!";     // 改变内容
}
</script>

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

</body>
</html>
```

# 改变 HTML 图像(4改变HTML图像.HTML)

```
<!DOCTYPE html>
<html>
<body>
<script>
function changeImage()
{
element=document.getElementById('myimage')
if (element.src.match("bulbon"))
   {
   element.src="/i/eg_bulboff.gif";
   }
else
   {
   element.src="/i/eg_bulbon.gif";
   }
}
</script>

<img id="myimage" onclick="changeImage()" src="/i/eg_bulboff.gif">

<p>点击灯泡来点亮或熄灭这盏灯</p>

</body>
</html>
```

# 改变 HTML样式.HTML(5改变 HTML样式.HTML)

```
<!DOCTYPE html>
<html>
<body>

<h1>我的第一段 JavaScript</h1>

<p id="demo">
JavaScript 能改变 HTML 元素的样式。
</p>

<script>
function myFunction()
{
x=document.getElementById("demo")  // 找到元素
x.style.color="#ff0000";           // 改变样式
}
</script>

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

</body>
</html>
```

# 验证输入.HTML(6验证输入.HTML)

```
<!DOCTYPE html>
<html>
<body>

<h1>我的第一段 JavaScript</h1>

<p>请输入数字。如果输入值不是数字，浏览器会弹出提示框。</p>

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

<script>
function myFunction()
{
var x=document.getElementById("demo").value;
if(x==""||isNaN(x))
          {
          alert("Not Numeric");
          }
}
</script>

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

</body>
</html>
```

# 外部的JAVASCRIPT.HTML(7外部的JAVASCRIPT.HTML)

```
<!DOCTYPE html>
<html>
<body>

<h1>My Web Page</h1>

<p id="demo">A Paragraph.</p>

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

<p><b>注释: </b>myFunction 保存在名为 "myScript.js" 的外部文件中。</p>

<script type="text/javascript" src="/js/myScript.js"></script>

</body>
</html>
```

- **JAVASCRIPT 对大小写敏感。**
- JAVASCRIPT 对大小写是敏感的。
- **JAVASCRIPT 注释**
- JAVASCRIPT 不会执行注释。
- 我们可以添加注释来对 JAVASCRIPT 进行解释，或者提高代码的可读性。
- 单行注释以 // 开头。
- **JAVASCRIPT 多行注释**
- 多行注释以 /* 开始，以 */ 结尾。

## JavaScript 变量 变量是存储信息的容器。

```
var x=2;
var y=3;
var z=x+y;
```

与代数一样，JavaScript 变量可用于存放值（比如 x=2）和表达式（比如 z=x+y）。

变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, totalvolume）。

- 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）

**提示：** JavaScript 语句和 JavaScript 变量都对大小写敏感。

# 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。

**例子**

```
var pi=3.14;
var name="Bill Gates";
var answer='Yes I am!';
```

# 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：

**实例**

```
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：

**实例**

```
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：

**实例**

```
var answer="Nice to meet you!";
var answer="He is called 'Bill'";
var answer='He is called "Bill"';
```

# JAVASCRIPT 布尔

布尔（逻辑）只能有两个值：true 或 false。

```
var x=true
var y=false
```

布尔常用在条件测试中。您将在本教程稍后的章节中学到更多关于条件测试的知识。

# JAVASCRIPT 数组(8JAVASCRIPT数组.HTML)

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):

**实例**

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

# JAVASCRIPT 对象(9JAVASCRIPT对象.HTML)

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:

**实例**

```
name=person.lastname;
name=person["lastname"];
```

# UNDEFINED 和 NULL(10UNDEFINED和NULL.HTML)

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：

**实例**

```
name=person.lastname;
name=person["lastname"];
```

# 创建 JAVASCRIPT 对象(11创建JAVASCRIPT对象.HTML)

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：

**实例**

```
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
```

本例使用 String 对象的 length 属性来查找字符串的长度：

```
var message="Hello World!";
var x=message.length;
```

在以上代码执行后，x 的值是：

```
12
```

# 访问对象的方法

您可以通过下面的语法调用方法：

```
objectName.methodName()
```

这个例子使用 String 对象的 toUpperCase() 方法来把文本转换为大写：

```
var message="Hello world!";
var x=message.toUpperCase();
```

在以上代码执行后，x 的值是：

```
HELLO WORLD!
```

# 调用带参数的函数(12调用带参数的函数.HTML)

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。

**实例**

```
<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(13带有返回值的函数.HTML)

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();
```

# 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 |

# JAVASCRIPT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 |
| != | 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 |

---

**如何使用**

可以在条件语句中使用比较运算符对值进行比较，然后根据结果来采取行动：

```
if (age<18) document.write("Too young");
```

# JAVASCRIPT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"。

# JAVASCRIPT条件语句

## 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

# JAVASCRIPT SWITCH 语句

**实例**

显示今日的周名称。请注意 Sunday=0, Monday=1, Tuesday=2, 等等：

```
var day=new Date().getDay();
switch (day)
{
case 0:
  x="Today it's Sunday";
  break;
case 1:
  x="Today it's Monday";
  break;
case 2:
  x="Today it's Tuesday";
  break;
case 3:
  x="Today it's Wednesday";
  break;
case 4:
  x="Today it's Thursday";
  break;
case 5:
  x="Today it's Friday";
  break;
case 6:
  x="Today it's Saturday";
  break;
}
```

x 的结果：

```
Today it's Wednesday
```

**default 关键词**

请使用 default 关键词来规定匹配不存在时做的事情：

**实例**

如果今天不是周六或周日，则会输出默认的消息：

```
var day=new Date().getDay();
switch (day)
{
case 6:
  x="Today it's Saturday";
  break;
case 0:
  x="Today it's Sunday";
  break;
default:
  x="Looking forward to the Weekend";
}
```

x 的结果：

```
Looking forward to the Weekend
```

# JAVASCRIPT 循环

```
<!DOCTYPE html>
<html>
<body>

<script>
cars=["BMW","Volvo","Saab","Ford"];
for (var i=0;i<cars.length;i++)
{
document.write(cars[i] + "<br>");
}
</script>

</body>
</html>
```

# JAVASCRIPT 循环

```
<!DOCTYPE html>
<html>
<body>

<p>点击下面的按钮，只要 i 小于 5 就一直循环代码块。</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>

<script>
function myFunction()
{
var x="",i=0;
while (i<5)
   {
   x=x + "The number is " + i + "<br>";
   i++;
   }
document.getElementById("demo").innerHTML=x;
}
</script>

</body>
</html>
```

# JAVASCRIPT 循环

```
<!DOCTYPE html>
<html>
<body>

<p>点击下面的按钮，只要 i 小于 5 就一直循环代码块。</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>

<script>
function myFunction()
{
var x="",i=0;
do
   {
   x=x + "The number is " + i + "<br>";
   i++;
   }
while (i<5)
document.getElementById("demo").innerHTML=x;
}
</script>

</body>
</html>
```

# JAVASCRIPT BREAK 语句

break 语句可用于跳出循环。

break 语句跳出循环后，会继续执行该循环之后的代码（如果有的话）：

**实例**

```
for (i=0;i<10;i++)
  {
  if (i==3)
    {
    break;
    }
  x=x + "The number is " + i + "<br>";
  }
```

# JAVASCRIPT CONTINUE 语句

continue 语句中断循环中的迭代，如果出现了指定的条件，然后继续循环中的下一个迭代。

该例子跳过了值 3:

**实例**

```
for (i=0;i<=10;i++)
 {
 if (i==3) continue;
  x=x + "The number is " + i + "<br>";
  }
```

# 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：

**实例**

```
cars=["BMW","Volvo","Saab","Ford"];
list:
{
document.write(cars[0] + "<br>");
document.write(cars[1] + "<br>");
document.write(cars[2] + "<br>");
break list;
document.write(cars[3] + "<br>");
document.write(cars[4] + "<br>");
document.write(cars[5] + "<br>");
}
```

# JAVASCRIPT 错误 – TRY 和 CATCH

**实例**

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>
```

# JAVASCRIPT 错误 – THROW(14JAVASCRIPT错误THROW.HTML)

```
<!DOCTYPE html>
<html>
<body>

<script>
function myFunction()
{
try
{
var x=document.getElementById("demo").value;
if(x=="")      throw "值为空";
if(isNaN(x)) throw "不是数字";
if(x>10)       throw "太大";
if(x<5)        throw "太小";
}
catch(err)
{
var y=document.getElementById("mess");
y.innerHTML="错误: " + err + "。";
}
}
</script>

<h1>我的第一个 JavaScript 程序</h1>
<p>请输入 5 到 10 之间的数字: </p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">测试输入值</button>
<p id="mess"></p>

</body>
</html>
```

# JAVASCRIPT改变 HTML 属性

如需改变 HTML 元素的属性，请使用这个语法：

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

**实例**

本例改变了 <img> 元素的 src 属性：

```
<!DOCTYPE html>
<html>
<body>

<img id="image" src="smiley.gif">

<script>
document.getElementById("image").src="landscape.jpg";
</script>

</body>
</html>
```

# JAVASCRIPT改变HTML样式(15JAVASCRIPT改变HTML样式.HTML)

如需改变 HTML 元素的样式，请使用这个语法：

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

**例子 1**

下面的例子会改变 <p> 元素的样式：

```
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color="blue";
</script>
```

# ONLOAD和ONUNLOAD事件(16ONLOAD和ONUNLOAD事件.HTML)

```
<!DOCTYPE html>
<html>
<body onload="checkCookies()">

<script>
function checkCookies()
{
if (navigator.cookieEnabled==true)
            {
            alert("已启用 cookie")
            }
else
            {
            alert("未启用 cookie")
            }
}
</script>

<p>提示框会告诉你，浏览器是否已启用 cookie。</p>
</body>
</html>
```

# ONCHANGE事件(17ONCHANGE事件.HTML)

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
var x=document.getElementById("fname");
x.value=x.value.toUpperCase();
}
</script>
</head>
<body>

请输入英文字符: <input type="text" id="fname" onchange="myFunction()">
<p>当您离开输入字段时，会触发将输入文本转换为大写的函数。</p>

</body>
</html>
```

# ONMOUSEOVER和ONMOUSEOUT事件

(18ONMOUSEOVER和ONMOUSEOUT事件.HTML)

```
<!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

(19ONMOUSEDOWN、ONMOUSEUP以及ONCLICK事件.HTML)

```
<!DOCTYPE html>
<html>
<body>

<div onmousedown="mDown(this)" onmouseup="mUp(this)" style="background-
color:green;color:#ffffff;width:90px;height:20px;padding:40px;font-size:12px;">请点击这里</div>

<script>
function mDown(obj)
{
obj.style.backgroundColor="#1ec5e5";
obj.innerHTML="请释放鼠标按钮"
}

function mUp(obj)
{
obj.style.backgroundColor="green";
obj.innerHTML="请按下鼠标按钮"
}
</script>

</body>
</html>
```

# 创建新的HTML元素(20创建新的HTML元素.HTML)

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。

**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 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>
```

# 删除已有的HTML元素(21删除已有的HTML元素.HTML)

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>
```

# 日期对象用于处理日期和时间(22日期对象用于处理日期和时间.HTML)

## JavaScript Date（日期）对象 实例

返回当日的日期和时间

如何使用 Date() 方法获得当日的日期。

getTime()

getTime() 返回从 1970 年 1 月 1 日至今的毫秒数。

setFullYear()

如何使用 setFullYear() 设置具体的日期。

toUTCString()

如何使用 toUTCString() 将当日的日期（根据 UTC）转换为字符串。

getDay()

如何使用 getDay() 和数组来显示星期，而不仅仅是数字。

显示一个钟表

如何在网页上显示一个钟表。

## 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");
}
```

# JAVASCRIPT数组合并(23JAVASCRIPT数组合并.HTML)

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

var arr2 = new Array(3)
arr2[0] = "James"
arr2[1] = "Adrew"
arr2[2] = "Martin"

document.write(arr.concat(arr2))

</script>

</body>
</html>
```

# 用数组的元素组成字符串JOIN (24用数组的元素组成字符串JOIN.HTML)

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(3);
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr.join());

document.write("<br />");

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

</script>

</body>
</html>
```

# 文字数组SORT(25文字数组SORT.HTML)

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
document.write(arr.sort())

</script>

</body>
</html>
```

# 数字数组SORT(26数字数组SORT.HTML)

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
document.write(arr.sort())

</script>

</body>
</html>
```

# JAVASCRIPT MATH（算数）对象

**实例**

round()

如何使用 round()。

random()

如何使用 random() 来返回 0 到 1 之间的随机数。

max()

如何使用 max() 来返回两个给定的数中的较大的数。

min()

如何使用 min() 来返回两个给定的数中的较小的数。

**算数值**

JavaScript 提供 8 种可被 Math 对象访问的算数值：

- 常数
- 圆周率
- 2 的平方根
- 1/2 的平方根
- 2 的自然对数
- 10 的自然对数
- 以 2 为底的 e 的对数
- 以 10 为底的 e 的对数

- Math.E
- Math.PI
- Math.SQRT2
- Math.SQRT1_2
- Math.LN2
- Math.LN10
- Math.LOG2E
- Math.LOG10E

# WINDOW HISTORY BACK FORWARD

history.back() 方法加载历史列表中的前一个 URL。

这与在浏览器中点击后退按钮是相同的：

**实例**

在页面上创建后退按钮：

```
<html>
<head>
<script>
function goBack()
  {
  window.history.back()
  }
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

history forward() 方法加载历史列表中的下一个 URL。

这与在浏览器中点击前进按钮是相同的：

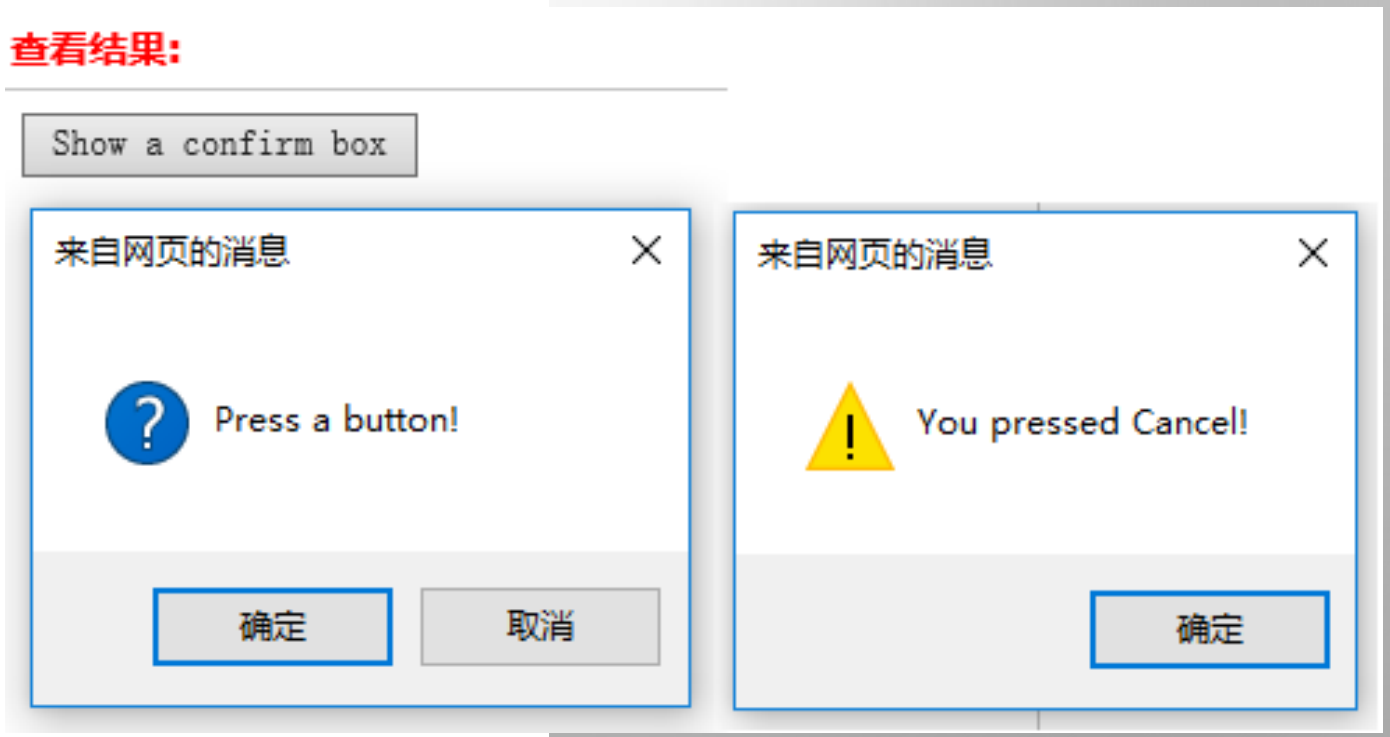
**实例**

在页面上创建一个向前的按钮：

```
<html>
<head>
<script>
function goForward()
  {
  window.history.forward()
  }
</script>
</head>
<body>

<input type="button" value="Forward" onclick="goForward()">

</body>
</html>
```

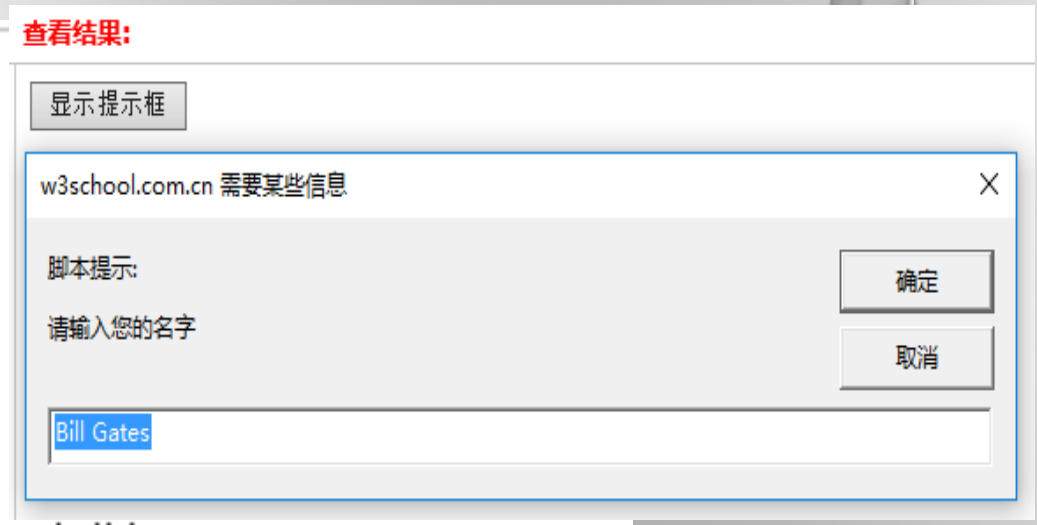
# JAVASCRIPT确认框(27JAVASCRIPT确认框.HTML)

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button!");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show a confirm box" />

</body>
</html>
```



# JAVASCRIPT提示框(28JAVASCRIPT提示框.HTML)

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
  {
  var name=prompt("请输入您的名字","Bill Gates")
  if (name!=null && name!="")
    {
    document.write("你好！ " + name + " 今天过得怎么样？")
    }
  }
</script>
</head>
<body>

<input type="button" onclick="disp_prompt()" value="显示提示框" />

</body>
</html>
```



# JAVASCRIPT简单的计时(29JAVASCRIPT简单的计时.HTML)

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 秒! ')",5000)
}
</script>
</head>

<body>
<form>
<input type="button" value="显示定时的警告框" onClick = "timedMsg()">
</form>
<p>请点击上面的按钮。警告框会在 5 秒后显示。</p>
</body>

</html>
```

# 带有停止按钮的无穷循环中的计时事件

(30带有停止按钮的无穷循环中的计时事件.HTML)

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}

function stopCount()
{
c=0;
setTimeout("document.getElementById('txt').value=0",0);
clearTimeout(t);
}
</script>
</head>
<body>
<form>
<input type="button" value="开始计时！" onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="停止计时！" onClick="stopCount()">
</form>

<p>请点击上面的"开始计时"按钮来启动计时器。输入框会一直进行计时，从 0 开始。点击"停止计时"按钮可以终止计时，并将计数重置为 0。</p>

</body>

</html>
```