JavaScript Study Notes

---

2 JavaScript 基础语法

- 2-2 变量命名

  变量名字可以任意取，只不过取名字要遵循一些规则:

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

```
正确:
    mysum
    _mychar
    $numa1
错误:
  6num  //开头不能用数字
  %sum //开头不能用除 (_ $) 外特殊符号，如 (%  + /等)
  sum+num //开头中间不能使用除 (_ $) 外特殊符号，如 (%  + /等)
```

  2. 变量名区分大小写，如:A 与 a 是两个不同变量。
  3. 不允许使用 JavaScript 关键字和保留字做变量名。

关键字			
break	else	new	var
case	finally	return	void
catch	for	switch	while
default	if	throw	
delete	in	try	
do	instanceof	typeof	

保留字			
abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

- 2-3 变量声明

  声明变量语法:

```
var 变量名;
```

var 就相当于找盒子的动作，在 JavaScript 中是关键字（即保留字），这个关键字的作用是声明变量，并为“变量”准备位置（即内存）。

```
var mynum ; //声明一个变量 mynum
```

当然，我们可以一次找一个盒子，也可以一次找多个盒子，所以 Var 还可以一次声明多个变量，变量之间用“,”逗号隔开。

```
var num1,mun2 ; //声明两个变量 num1 和 num2
```

注意：变量也可以不声明，直接使用，但为了规范，需要先声明，后使用。

- 2-4 变量赋值

使用 `"="` 号给变量存储内容，看下面的语句：

```
var mynum = 5 ; //声明变量 mynum 并赋值。
```

也可以这样写：

```
var mynum; //声明变量 mynum
mynum = 5 ; //给变量 mynum 赋值
```

注：这里 `"="` 号的作用是给变量赋值，不是等于号。

变量是无所不能的容器，你可以把任何东西存储在变量里，如数值、字符串、布尔值等，例如：

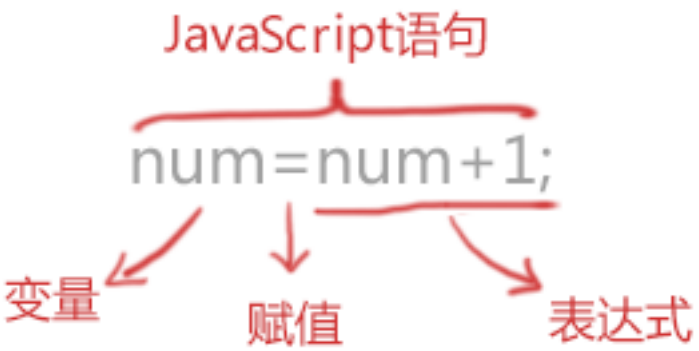
```
var num1 = 123;        // 123 是数值
var num2 = " 一二三";     //" 一二三" 是字符串
var num3=true;     //布尔值 true（真），false(假)
```

其中，num1 变量存储的内容是数值；num2 变量存储的内容是字符串，字符串需要用一对引号 `""` 括起来，num3 变量存储的内容是布尔值（true、false）。

- 2-5 表达式

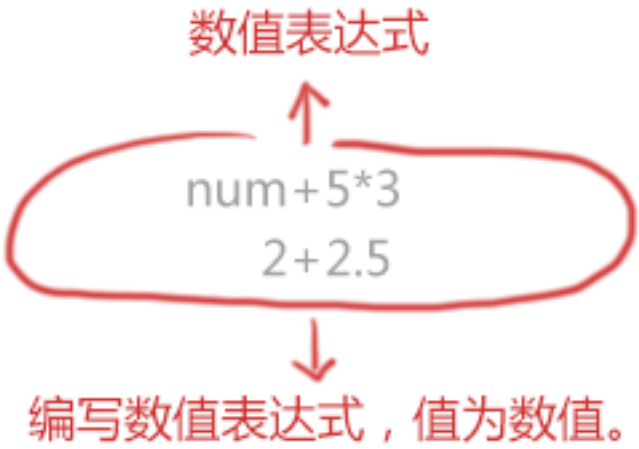
表达式是指具有一定的值、用操作符把常数和变量连接起来的代数式。一个表达式可以包含常数或变量。

我们先看看下面的 JavaScript 语句：

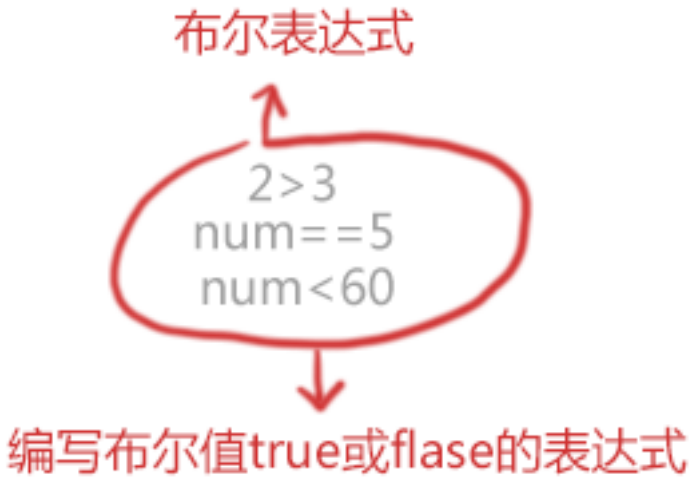


生活中“再见”表达方法很多，如：英语（goodbye）、网络语（88）、肢体语（挥挥手）等。在 JavaScript 表达式无处不在，所以一定要知道可以表达哪些内容，看看下面几种情况：

注意：串表达式中 mychar 是变量



注意：数值表达式中 num 是变量



注意：布尔表达式中 num 是变量

- 2-6 操作符

  操作符是用于在 JavaScript 中指定一定动作的符号。

  (1) 操作符

  看下面这段 JavaScript 代码。

```
sum = numa + numb;
```

  其中的 `"="` 和 `"+"` 都是操作符。

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

  注意：“=”操作符是赋值，不是等于。

  (2) `"+"` 操作符

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

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

- 2-7 ++ 和–

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

```
mynum = 10;
mynum++; //mynum 的值变为 11
mynum--; //mynum 的值又变回 10
```

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

```
mynum = mynum + 1;//等同于 mynum++
mynum = mynum - 1;//等同于 mynum--
```

- 2-8 比较操作符

  在 JavaScript 中，这样的比较操作符有很多，这些操作符的含义如下：

操作符	描述
<	小于
>	大于
<=	小于或等于
>=	大于或等于
==	等于
!=	不等于

看看下面例子:

```
var a = 5;//定义 a 变量，赋值为 5
var b = 9; //定义 b 变量，赋值为 9
document.write (a<b); //a 小于 b 的值吗？结果是真 (true)
document.write (a>=b); //a 大于或等于 b 的值吗？结果是假 (false)
document.write (a!=b); //a 不等于 b 的值吗？结果是真 (true)
document.write (a==b); //a 等于 b 的值吗？结果是假 (false)
```

- 2-9 逻辑操作符
  - “&&” 是逻辑与操作符，只有“&&”两边值同时满足 (同时为真)，整个表达式值才为真。
  - "||" 逻辑或操作符，当两个条件中有任一个条件满足，“逻辑或”的运算结果就为“真”。
  - "!" 是逻辑非操作符，也就是“不是”的意思，非真即假，非假即真。
- 2-12 操作符优先级

  操作符之间的优先级（高到低）：

  算术操作符 → 比较操作符 → 逻辑操作符 → “=” 赋值符号

  如果同级的运算是按从左到右次序进行，多层括号由里向外。

3 数组

- 3-1 数组

  数组是一个值的集合，每个值都有一个索引号，从 0 开始，每个索引都有一个相应的值，根据需要添加更多数值。

```
<script type="text/javascript">
 var myarr=new Array(); //定义数组
 myarr[0]=80;
 myarr[1]=60;
 myarr[2]=99;
 document.write(" 第一个人的成绩是:"+myarr[0]);
 document.write(" 第二个人的成绩是:"+myarr[1]);
 document.write(" 第三个人的成绩是:"+myarr[2]);
</script>
```

- 3-2 创建数组

  使用数组之前首先要创建，而且需要把数组本身赋至一个变量。

  创建数组语法:

```
var myarray=new Array();
```

我们创建数组的同时，还可以为数组指定长度，长度可任意指定。

```
var myarray= new Array(8); //创建数组，存储 8 个数据。
```

注意：

1. 创建的新数组是空数组，没有值，如输出，则显示 undefined。
2. 虽然创建数组时，指定了长度，但实际上数组都是变长的，也就是说即使指定了长度为 8，仍然可以将元素存储在规定长度以外。

- 3-3 数组赋值
  - 第一种方法：

    ```
    var myarray=new Array(); //创建一个新的空数组
    myarray[0]=66; //存储第 1 个人的成绩
    myarray[1]=80; //存储第 2 个人的成绩
    ```

  - 第二种方法：

    ```
    var myarray = new Array(66,80,90,77,59);//创建数组同时赋值
    ```

  - 第三种方法：

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

  注意：

  1. 数组存储的数据可以是任何类型（数字、字符、布尔值等）
  2. 数组每个值有一个索引号，从 0 开始。

- 3-4 向数组增加一个新元素

  只需使用下一个未用的索引，任何时刻可以不断向数组增加新元素。

- 3-5 使用数组元素

要得到一个数组元素的值，只需引用数组变量并提供一个索引，如：第一个人的成绩表示方法：`myarray[0]` 第三个人的成绩表示方法：`myarray[2]`

- 3-6 数组属性

  Length 属性表示数组的长度，即数组中元素的个数。

  语法：

```
myarray.length; //获得数组 myarray 的长度
```

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

```
var arr=[55,32,5,90,60,98,76,54];//包含 8 个数值的数组 arr
document.write(arr.length); //显示数组长度 8
document.write(arr[7]); //显示第 8 个元素的值 54
```

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

```
arr.length=10; //增大数组的长度
document.write(arr.length); //数组长度已经变为 10
```

  数组随元素的增加，长度也会改变，如下：

```
var arr=[98,76,54,56,76]; // 包含 5 个数值的数组
document.write(arr.length); //显示数组的长度 5
arr[15]=34;  //增加元素，使用索引为 15，赋值为 34
alert(arr.length); //显示数组的长度 16
```

- 3-7 二维数组

  二维数组的表示：

```
myarray[ ][ ]
```

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

  1. 二维数组的定义方法一

```
var myarr=new Array();  //先声明一维
for(var i=0;i<2;i++){   //一维长度为 2
   myarr[i]=new Array();  //再声明二维
   for(var j=0;j<3;j++){   //二维长度为 3
   myarr[i][j]=i+j;    // 赋值，每个数组元素的值为 i+j
   }
 }
```

  2. 二维数组的定义方法二

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

  3. 赋值

```
myarr[0][1]=5; //将 5 的值传入到数组中，覆盖原有值。
```

4 流程控制语句

- 4-1 if 语句

  if 语句是基于条件成立才执行相应代码时使用的语句。

  语法：

```
if(条件) {
  条件成立时执行代码
}
```

注意：**if** 小写，大写字母（**IF**）会出错!

- 4-2 if...else 语句

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

  语法:

```
if(条件)
{ 条件成立时执行的代码}
else
{条件不成立时执行的代码}
```

- 4-3 if...else 嵌套语句

  要在多组语句中选择一组来执行，使用 if..else 嵌套语句。

  语法:

```
if(条件 1)
{ 条件 1 成立时执行的代码}
else  if(条件 2)
{ 条件 2 成立时执行的代码}
...
else  if(条件 n)
{ 条件 n 成立时执行的代码}
else
{ 条件 1、2 至 n 不成立时执行的代码}
```

- 4-4 switch 语句

  当有很多种选项的时候，switch 比 if else 使用更方便。

  语法:

```
switch(表达式)
{
case 值 1:
  执行代码块 1
  break;
case 值 2:
  执行代码块 2
  break;
...
case 值 n:
  执行代码块 n
  break;
default:
  与 case 值 1 、 case 值 2...case 值 n 不同时执行的代码
}
```

  语法说明:

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

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

- 4-5 for 循环

  for 语句结构：

```
for(初始化变量;循环条件;循环迭代)
{
    循环语句
 }
```

- 4-6 while 循环

  while 语句结构：

```
while(判断条件)
{
    循环语句
 }
```

- 4-7 do…while 循环

  while 语句结构：

```
while(判断条件)
{
    循环语句
 }
```

- 4-8 退出循环 break

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

  格式如下：

```
for(初始条件;判断条件;循环后条件值更新)
{
  if(特殊情况)
  {break;}
  循环代码
}
```

  当遇到特殊情况的时候，循环就会立即结束。

- 4-9 继续循环 continue

  continue 的作用是仅仅跳过本次循环，而整个循环体继续执行。

  语句结构：

```
for(初始条件;判断条件;循环后条件值更新)
{
  if(特殊情况)
  { continue; }
 循环代码
}
```

  上面的循环中，当特殊情况发生的时候，本次循环将被跳过，而后续的循环则不会受到影响。

5 函数

- 5-2 定义函数

  如何定义一个函数呢？看看下面的格式：

```
function  函数名( )
{
     函数体;
}
```

function 定义函数的关键字，“函数名”你为函数取的名字，“函数体”替换为完成特定功能的代码。

- 5-3 函数调用

  函数定义好后，是不能自动执行的，需要调用它，直接在需要的位置写函数名。

  第一种情况：在 `<script>` 标签内调用。

```
  <script type="text/javascript">
    function add2()
    {
         sum = 1 + 1;
         alert(sum);
    }
  add2();//调用函数，直接写函数名。
</SCRIPT>
```

  第二种情况：在 HTML 文件中调用，如通过点击按钮后调用定义好的函数。

```
<html>
<head>
<script type="text/javascript">
   function add2()
   {
         sum = 5 + 6;
         alert(sum);
   }
</script>
</head>
<body>
<form>
<input type="button" value="click it" onclick="add2()">  //按钮,onclick 点击事件，直接写函数名
</form>
</body>
</html>
```

- 5-4 有参数的函数

  定义函数还可以如下格式：

```
function 函数名(参数 1,参数 2)
{
     函数代码
}
```

  注意：参数可以多个，根据需要增减参数个数。参数之间用（逗号,）隔开。

- 5-5 返回值的函数

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

  我们只要把“document.write(sum)”这行改成如下代码：

```
function add2(x,y)
{
```

```
    sum = x + y;
    return sum; //返回函数值,return 后面的值叫做返回值。
}
```

还可以通过变量存储调用函数的返回值，代码如下:

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

注意：函数中参数和返回值不只是数字，还可以是字符串等其它类型。

6 事件响应，让网页交互

- 6-1 什么是事件?

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

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

  主要事件表:

事件	说明
onclick	鼠标单击事件
onmouseover	鼠标经过事件
onmouseout	鼠标移开事件
onchange	文本框内容改变事件
onselect	文本框内容被选中事件
onfocus	光标聚集
onblur	光标离开
onload	网页导入
onunload	关闭网页

- 6-2 鼠标单击事件 (onclick)

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

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

```
<html>
<head>
   <script type="text/javascript">
      function add2(){
        var numa,numb,sum;
        numa=6;
        numb=8;
```

```
        sum=numa+numb;
        document.write(" 两数和为:"+sum);  }
   </script>
</head>
<body>
   <form>
      <input name="button" type="button" value=" 点击提交" onclick="add2()" />
   </form>
</body>
</html>
```

注意：在网页中，如使用事件，就在该元素中设置事件属性。

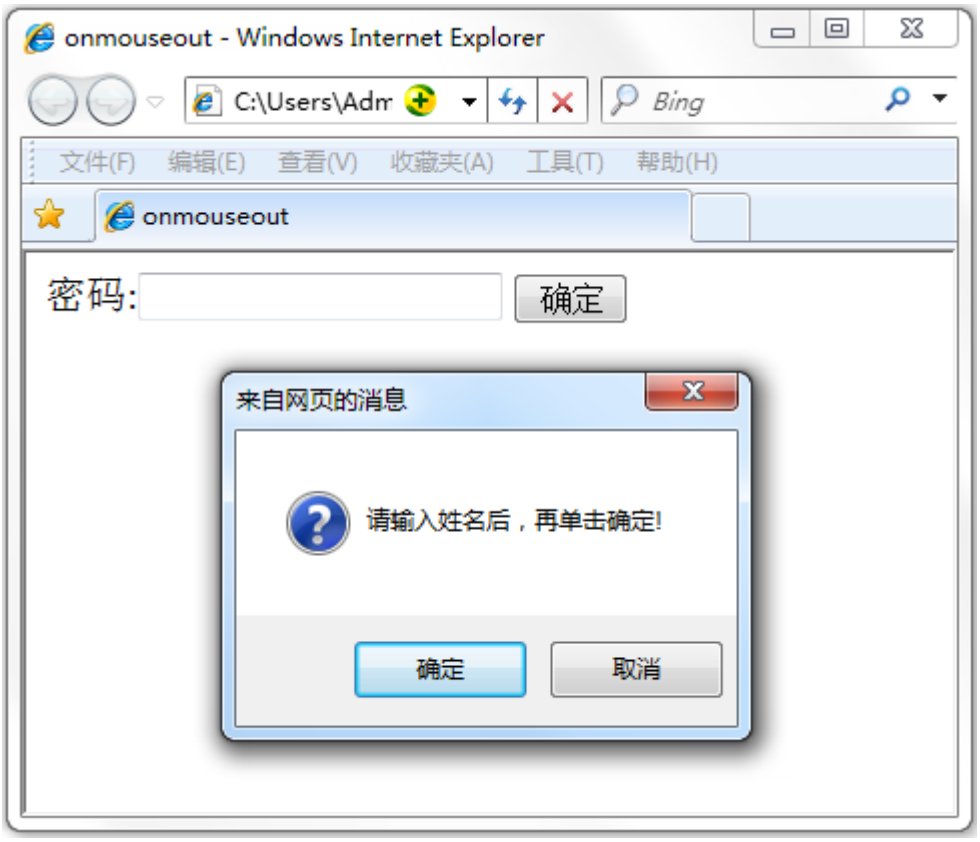
- 6-3 鼠标经过事件 (onmouseover)

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

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

```
<html>
<head>
   <script type="text/javascript">
      function info(){
        comfirm(" 请输入姓名后，再单击确定！")
          }
   </script>
</head>
<body>
   <form>
     密码: <input name="password" type="password">
      <input name="button" type="button" value=" 确定" onmouseover="info()" />
      <!--当鼠标经过“确定”按钮时，触发 onmouseover="info()"-->
   </form>
</body>
</html>
```

运行结果：



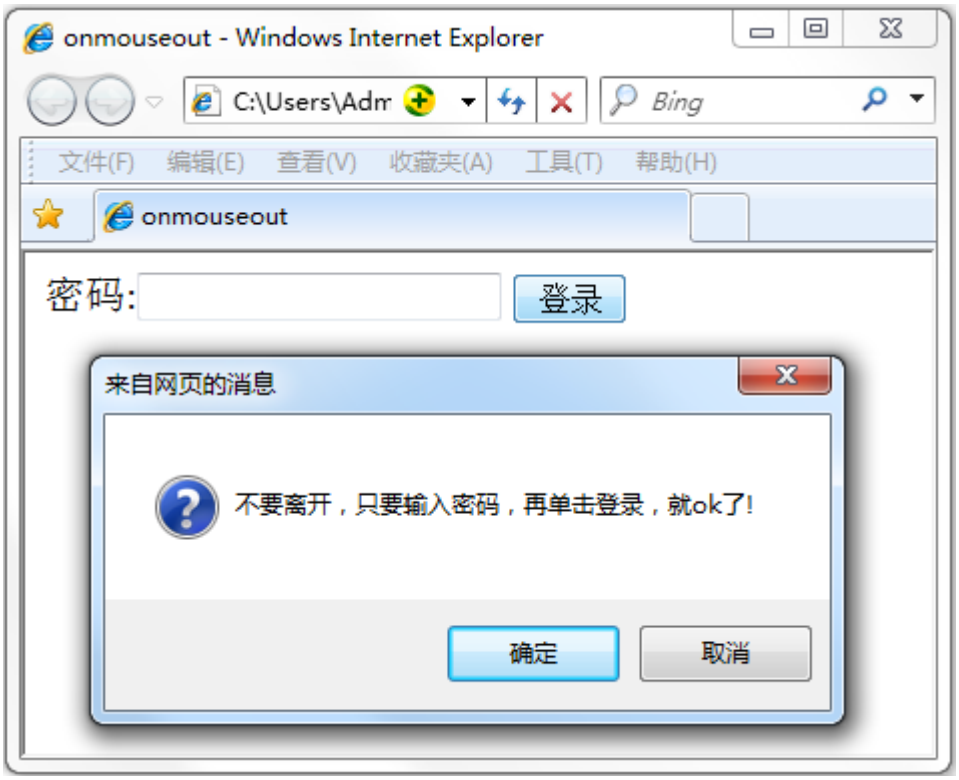
- 6-4 鼠标移开事件 (onmouseout)

  鼠标移开事件，当鼠标移开当前对象时，执行 onmouseout 调用的程序。

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

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>鼠标移开事件</title>
<script type="text/javascript">
  function message(){
    alert(" 不要离开，只要输入密码，再单击登录，就 ok 了"); }
</script>
</head>
<body>
<form>
  密码: <input name="password" type="password">
  <input name="button" type="button" value=" 登录" onmouseover="message()" />
      <!--当移开“登录”按钮时，触发 onmouseout="message()"-->
</form>
</body>
</html>
```

  运行结果:



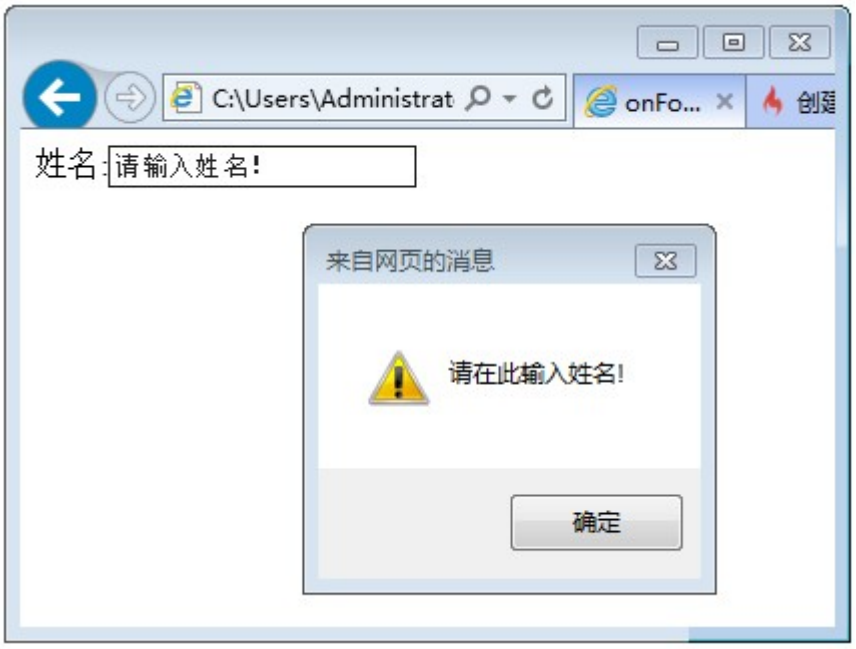
- 6-5 光标聚焦事件 (onfocus)

  当网页中的对象获得聚点时，执行 onfocus 调用的程序就会被执行。

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

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>光标聚焦事件</title>
<script type="text/javascript">
  function message(){
    alert(" 请在此输入姓名! "); }
</script>
</head>
<body>
<form>
  姓名: <input name="username" type="text" value=" 请输入姓名! " onfocus="message()">
  <!--当光标在文本框内（即文本框得到焦点）时，调用"message()"函数-->
</form>
</body>
</html>
```

  运行结果:



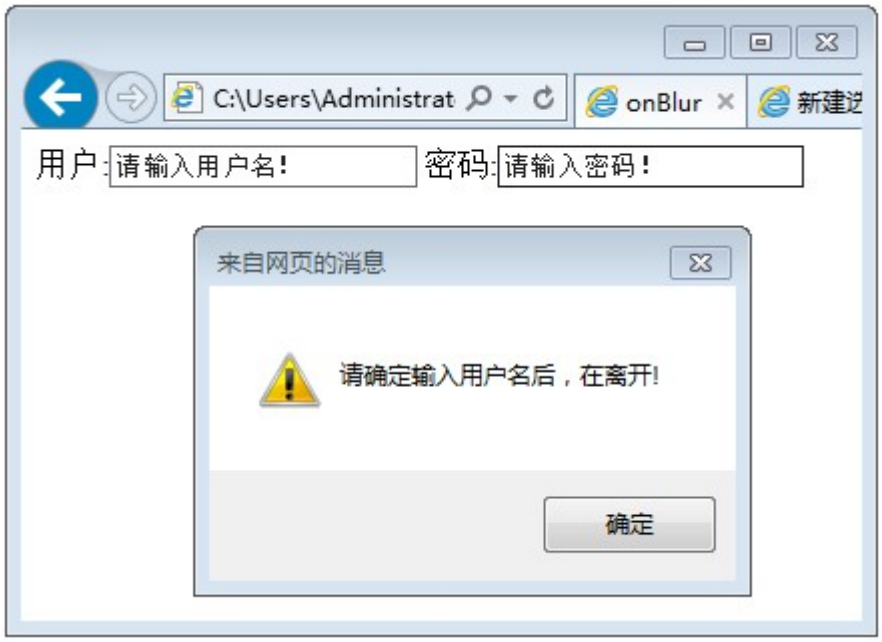
- 6-6 光标失焦事件 (onblur)

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

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

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>光标失焦事件</title>
<script type="text/javascript">
  function message(){
    alert(" 请确认已输入用户名后，再离开！"); }
</script>
</head>
<body>
<form>
  用户: <input name="username" type="text" value=" 请输入用户名！ " onblur="message()">
  密码: <input name="password" type="text" value=" 请输入密码！ ">
</form>
</body>
</html>
```

  运行结果：



- 6-7 内容选中事件 (onselect)

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

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

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>onselect</title>
<script type="text/javascript">
  function message(){
    alert(" 你触发了选中事件! "); }
</script>
</head>
<body>
<form>
  用户: <input name="username" type="text" value=" 请输入用户名! " onselect="message()">
  <!--当选中用户文本框内的文字时，触发 onselect 事件-->
</form>
</body>
</html>
```

  运行结果:

- 6-8 文本框内容改变事件 (onchange)

  通过改变文本框的内容来触发 onchange 事件，同时执行被调用的程序。

  如下代码，当用户将文本框内的文字改变后，弹出对话框“您改变了文本内容！”。

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>onchange</title>
```

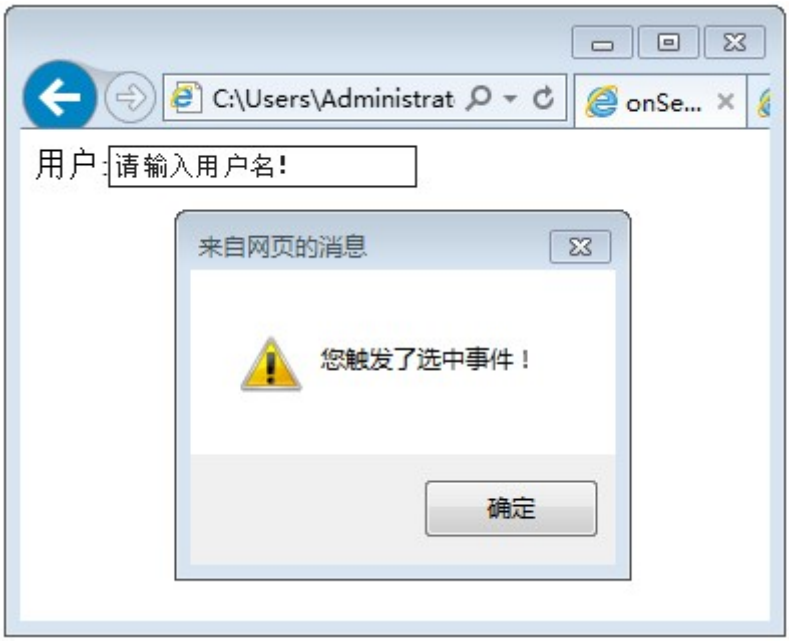
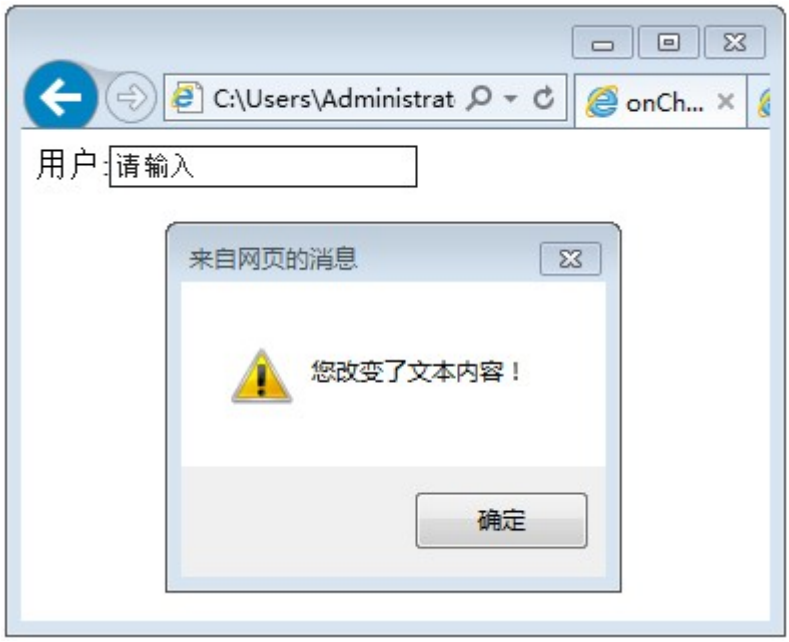


Figure 1: img

```
<script type="text/javascript">
  function message(){
    alert(" 你改变了文本内容! "); }
</script>
</head>
<body>
<form>
  用户: <input name="username" type="text" value=" 请输入用户名! " onchange="message()">
  <!--当改变用户文本框内的文字时，触发 onchange 事件-->
</form>
</body>
</html>
```

运行结果:



- 6-9 加载事件 (onload)

  事件会在页面加载完成后，立即发生，同时执行被调用的程序。注意：

  1. 加载页面时，触发 onload 事件，事件写在

     标签内。

  2. 此节的加载页面，可理解为打开一个新页面时。如下代码，当加载一个新页面时，弹出对话框“加载中，请稍等...”。

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>onload</title>
<script type="text/javascript">
  function message(){
    alert(" 加载中，请稍等..."); }
</script>
</head>
<body onchange="message()">
欢迎学习 JavaScript。
</body>
</html>
```

  运行结果：

- 6-10 卸载事件 (onunload)

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

  注意：不同浏览器对 **onunload** 事件支持不同。

  如下代码，当退出页面时，弹出对话框“您确定离开该网页吗？”。

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

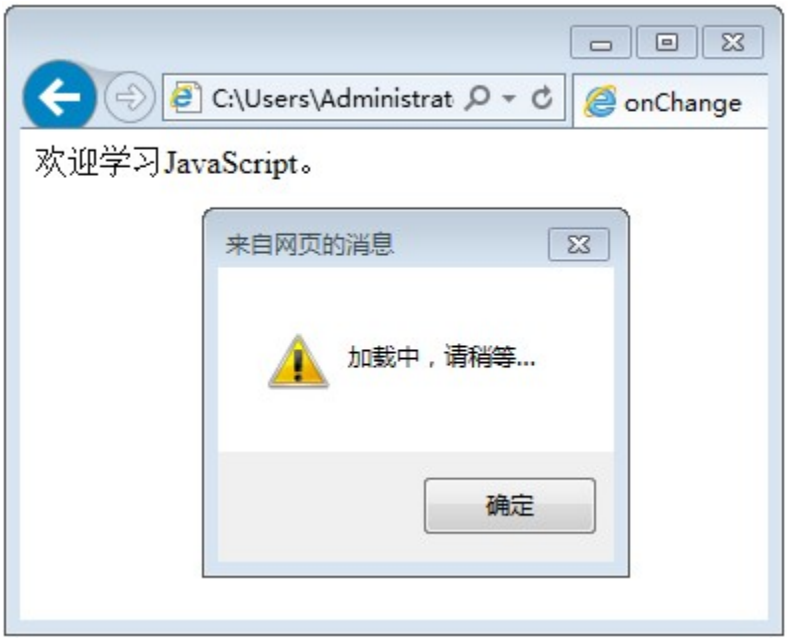
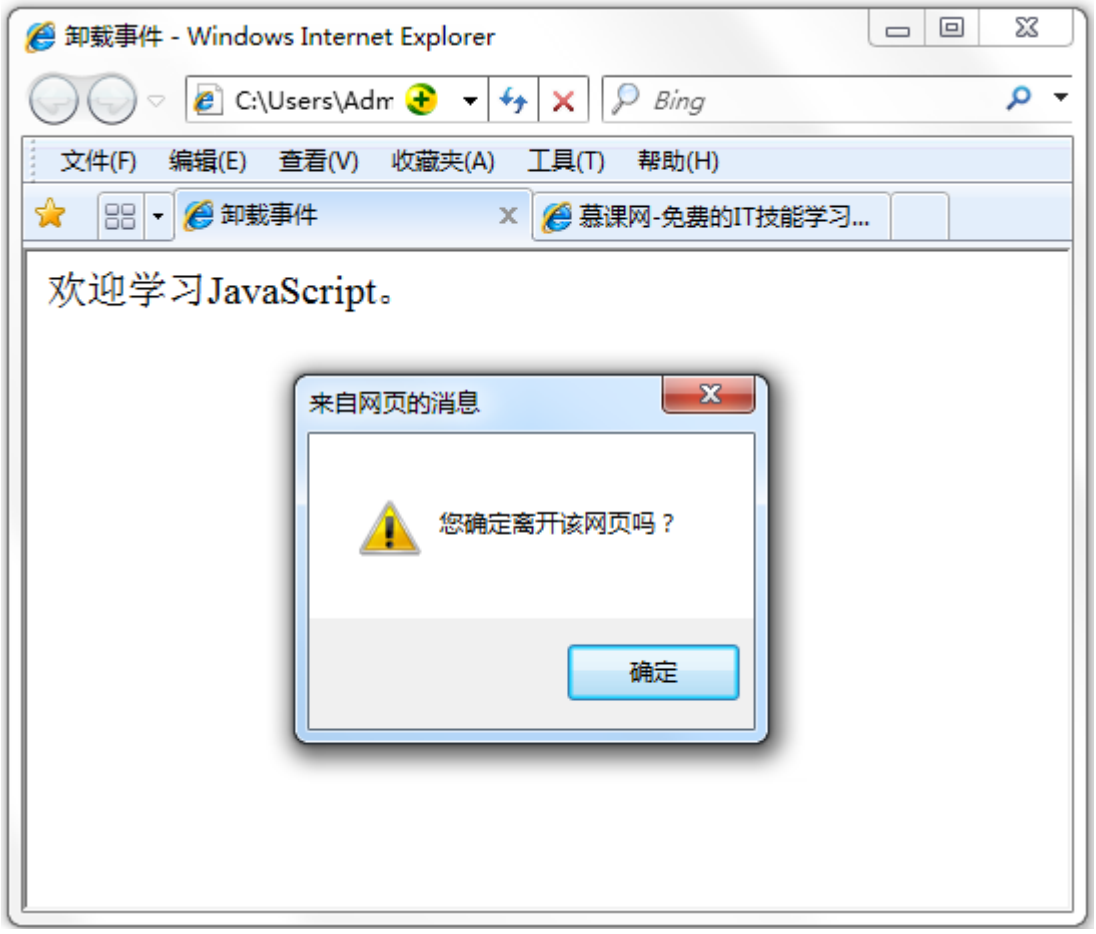


Figure 2: img

```
<title>onload</title>
<script type="text/javascript">
  window.onunload=onunload_message;
  function onunload_message(){
    alert(" 你确定离开该网页吗? "); }
</script>
</head>
<body>
欢迎学习 JavaScript。
</body>
</html>
```

运行结果：（**IE** 浏览器）



- 6-11 编程练习

  使用 JS 完成一个简单的计算器功能。实现 2 个输入框中输入整数后，点击第三个输入框能给出 2 个整数的加减乘除。

  提示：获取元素的值设置和获取方法为：例：赋值：document.getElementById(“id”) .value = 1；取值：var = document.getElementById(“id”) .value;

  注意：使用 parseInt() 函数可解析一个字符串，并返回一个整数。

```
<!DOCTYPE html>
<html>
 <head>
  <title> 事件</title>
  <script type="text/javascript">
   function count(){

    //获取第一个输入框的值
    var num1 = document.getElementById("txt1").value;
  //获取第二个输入框的值
  var num2 = document.getElementById("txt2").value;
  //获取选择框的值
  var op = document.getElementById("select").value;

  var res;

  //获取通过下拉框来选择的值来改变加减乘除的运算法则
```

```
  switch(op) {
      case "+":
          res = parseInt(num1) + parseInt(num2);  // 不然会是两个 String 的整合，e.g. 4 + 5 = 45
          break;
      case "-":
          res = num1 - num2;
          break;
        case "*":
          res = num1 * num2;
          break;
      case "/":
          res = num1 / num2;
  }
    //设置结果输入框的值
    document.getElementById("fruit").value = res;

   }
  </script>
 </head>
 <body>
   <input type='text' id='txt1' />
   <select id='select'>
      <option value='+'>+</option>
      <option value="-">-</option>
      <option value="*">*</option>
      <option value="/">/</option>
   </select>
   <input type='text' id='txt2' />
   <input type='button' value=' = ' onclick="count()"/> <!--通过 = 按钮来调用创建的函数，得到结果-->
   <input type='text' id='fruit' />
 </body>
</html>
```

7 JavaScript 内置对象

- 7-1 什么是对象

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

  对象的属性：反映该对象某些特定的性质的，如：字符串的长度、图像的长宽等；

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

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

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

  访问对象属性的语法：

```
objectName.propertyName
```

  如使用 Array 对象的 length 属性来获得数组的长度：

```
var myarray=new Array(6);//定义数组对象
var myl=myarray.length;//访问数组长度 length 属性
```

以上代码执行后，**myl** 的值将是：**6**

访问对象的方法：

```
objectName.methodName()
```

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

```
var mystr="Hello world!";//创建一个字符串
var request=mystr.toUpperCase(); //使用字符串对象方法
```

以上代码执行后，request 的值是：**HELLO WORLD!**

- 7-2 Date 日期对象

日期对象可以储存任意一个日期，并且可以精确到毫秒数（1/1000 秒）。

定义一个时间对象：

```
var Udate=new Date();
```

注意：使用关键字 new，Date() 的首字母必须大写。

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

如果要自定义初始值，可以用以下方法：

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

我们最好使用下面介绍的“方法”来严格定义时间。

访问方法语法：“.”

Date 对象中处理时间和日期的常用方法：

方法名称	功能描述
get/setDate()	返回/设置日期
get/setFullYear()	返回/设置年份，用四位数表示
get/setYear()	返回/设置年份
get/setMonth()	返回/设置月份。 0:一月...11:十二月，所以加一。
get/setHours()	返回/设置小时，24 小时制。
get/setMinutes()	返回/设置分钟数
get/setSeconds()	返回/设置秒钟数
get/setTime()	返回/设置时间（毫秒为单位）

  - 返回/设置年份方法

    `get/setFullYear()` 返回/设置年份，用四位数表示。

```
var mydate=new Date();//当前时间 2014 年 3 月 6 日
document.write(mydate+"<br>");//输出当前时间
document.write(mydate.getFullYear()+"<br>");//输出当前年份
```

```
mydate.setFullYear(81); //设置年份
document.write(mydate+"<br>"); //输出年份被设定为 0081 年。
```

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

结果：

```
Thu Mar 06 2014 10:57:47 GMT+0800
2014
Thu Mar 06 0081 10:57:47 GMT+0800
```

注意：

1. 结果格式依次为：星期、月、日、年、时、分、秒、时区。(火狐浏览器)
2. 不同浏览器，时间格式有差异。

– 返回星期方法

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

```
<script type="text/javascript">
  var mydate=new Date();//定义日期对象
  var weekday=[" 星期日"," 星期一"," 星期二"," 星期三"," 星期四"," 星期五"," 星期六"];
//定义数组对象，给每个数组项赋值
  var mynum=mydate.getDay();//返回值存储在变量 mynum 中
  document.write(mydate.getDay());//输出 getDay() 获取值
  document.write(" 今天是："+ weekday[mynum]);//输出星期几
</script>
```

注意：以上代码是在 2014 年 3 月 7 日，星期五运行。

结果：

```
5
今天是：星期五
```

– 返回/设置时间方法

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

如果将目前日期对象的时间推迟 1 小时，代码如下：

```
<script type="text/javascript">
  var mydate=new Date();
  document.write(" 当前时间："+mydate+"<br>");
  mydate.setTime(mydate.getTime() + 60 * 60 * 1000);
  document.write(" 推迟一小时时间：" + mydate);
</script>
```

结果：

```
当前时间：Thu Mar 6 11:46:27 UTC+0800 2014
推迟一小时时间：Thu Mar 6 12:46:27 UTC+0800 2014
```

注意：

1. 一小时 60 分，一分 60 秒，一秒 1000 毫秒
2. 时间推迟 1 小时，就是：“x.setTime(x.getTime() + 60 * 60 * 1000);”

- 7-6 String 字符串对象

定义字符串的方法就是直接赋值。比如：

```
var mystr = "I love JavaScript!"
```

定义 mystr 字符串后，我们就可以访问它的属性和方法。

访问字符串对象的属性 **length:**

stringObject.length; 返回该字符串的长度。

```
var mystr="Hello World!";
var myl=mystr.length;
```

以上代码执行后，myl 的值将是：12

访问字符串对象的方法：

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

```
var mystr="Hello world!";
var mynum=mystr.toUpperCase();
```

以上代码执行后，mynum 的值是：HELLO WORLD!

- 返回指定位置的字符

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

  语法：

  ```
  stringObject.charAt(index)
  ```

  参数说明：

参数	描述
index	必需。表示字符串中某个位置的数字，即字符在字符串中的下标。

  注意：

  1. 字符串中第一个字符的下标是 0。最后一个字符的下标为字符串长度减一（string.length-1）。
  2. 如果参数 index 不在 0 与 string.length-1 之间，该方法将返回一个空字符串。

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

  ```
  <script type="text/javascript">
    var mystr="I love JavaScript!"
    document.write(mystr.charAt(2));
  </script>
  ```

  注意：一个空格也算一个字符。

  以上代码的运行结果：

  ```
  l
  ```

- 返回指定的字符串首次出现的位置

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

  语法

```
stringObject.indexOf(substring, startpos)
```

参数说明：

参数	描述
substring	必需。规定需检索的字符串值。
startpos	可选的整数参数。规定在字符串中开始检索的位置。它的合法取值是 0 到 stringObject.length - 1。如省略该参数，则将从字符串的首字符开始检索。

说明：

1. 该方法将从头到尾地检索字符串 stringObject，看它是否含有子串 substring。
2. 可选参数，从 stringObject 的 startpos 位置开始查找 substring，如果没有此参数将从 stringObject 的开始位置查找。
3. 如果找到一个 substring，则返回 substring 的第一次出现的位置。stringObject 中的字符位置是从 0 开始的。

注意：

1. indexOf() 方法区分大小写。
2. 如果要检索的字符串值没有出现，则该方法返回 -1。

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

```
<script type="text/javascript">
  var str="I love JavaScript!"
  document.write(str.indexOf("I") + "<br />");
  document.write(str.indexOf("v") + "<br />");
  document.write(str.indexOf("v",8));
</script>
```

以上代码的输出：

```
0
4
9
```

– 字符串分割 split()

split() 方法将字符串分割为字符串数组，并返回此数组。

语法：

```
stringObject.split(separator,limit)
```

参数说明：

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

我们将按照不同的方式来分割字符串：

使用指定符号分割字符串，代码如下：

```
var mystr = "www.imooc.com";
document.write(mystr.split(".")+"<br>");
document.write(mystr.split(".", 2)+"<br>");
```

参数	描述
separator	必需。从该参数指定的地方分割 stringObject。
limit	可选参数，分割的次数，如设置该参数，返回的子串不会多于这个参数指定的数组，如果无此参数为不限制次数

Figure 3: img

运行结果：

```
www,imooc,com
www,imooc
```

将字符串分割为字符，代码如下：

```
document.write(mystr.split("")+"<br>");
document.write(mystr.split("", 5));
```

运行结果：

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

- 提取字符串 substring()

substring() 方法用于提取字符串中介于两个指定下标之间的字符。

语法：

```
stringObject.substring(startPos,stopPos)
```

参数说明：

参数	描述
startPos	必需。一个非负的整数，开始位置。
stopPos	可选。一个非负的整数，结束位置，如果省略该参数，那么返回的子串会一直到字符串对象的结尾。

注意：

1. 返回的内容是从 start 开始（包含 start 位置的字符）到 stop-1 处的所有字符，其长度为 stop 减 start。
2. 如果参数 start 与 stop 相等，那么该方法返回的就是一个空串（即长度为 0 的字符串）。
3. 如果 start 比 stop 大，那么该方法在提取子串之前会先交换这两个参数。

使用 substring() 从字符串中提取字符串，代码如下：

```
<script type="text/javascript">
  var mystr="I love JavaScript";
  document.write(mystr.substring(7));
  document.write(mystr.substring(2,6));
</script>
```

运行结果：

```
JavaScript
love
```

- 提取指定数目的字符 substr()

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

  语法：

```
stringObject.substr(startPos,length)
```

  参数说明：

参数	描述
startPos	必需。要提取的子串的起始位置。必须是数值。
length	可选。提取字符串的长度。如果省略，返回从 stringObject 的开始位置 *startPos* 到 stringObject 的结尾的字符。

  注意：

  1. 如果参数 startPos 是负数，从字符串的尾部开始算起的位置。也就是说，-1 指字符串中最后一个字符，-2 指倒数第二个字符，以此类推。
  2. 如果 startPos 为负数且绝对值大于字符串长度，startPos 为 0。

  使用 substr() 从字符串中提取一些字符，代码如下：

```
<script type="text/javascript">
  var mystr="I love JavaScript!";
  document.write(mystr.substr(7));
  document.write(mystr.substr(2,4));
</script>
```

  运行结果：

```
JavaScript!
love
```

- 7-12 Math 对象

  Math 对象，提供对数据的数学计算。

  使用 Math 的属性和方法，代码如下：

```
<script type="text/javascript">
  var mypi=Math.PI;
  var myabs=Math.abs(-15);
  document.write(mypi);
  document.write(myabs);
</script>
```

  运行结果：

```
3.141592653589793
15
```

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

  Math 对象属性

属性	说明
E	返回算术常量 e，即自然对数的底数（约等于 2.718）。
LN2	返回 2 的自然对数（约等于 0.693）。
LN10	返回 10 的自然对数（约等于 2.302）。
LOG2E	返回以 2 为底的 e 的对数（约等于 1.442）。
LOG10E	返回以 10 为底的 e 的对数（约等于 0.434）。
PI	返回圆周率（约等于 3.14159）。
SQRT1_2	返回返回 2 的平方根的倒数（约等于 0.707）。
SQRT2	返回 2 的平方根（约等于 1.414）。

Math 对象方法

方法	描述
abs(x)	返回数的绝对值。
acos(x)	返回数的反余弦值。
asin(x)	返回数的反正弦值。
atan(x)	返回数字的反正切值
atan2(y,x)	返回由 x 轴到点(x,y)的角度(以弧度为单位)
ceil(x)	对数进行上舍入。
cos(x)	返回数的余弦。
exp(x)	返回 e 的指数。
floor(x)	对数进行下舍入。
log(x)	返回数的自然对数（底为e）。
max(x,y)	返回 x 和 y 中的最高值。
min(x,y)	返回 x 和 y 中的最低值。
pow(x,y)	返回 x 的 y 次幂。
random()	返回 0 ~ 1 之间的随机数。
round(x)	把数四舍五入为最接近的整数。
sin(x)	返回数的正弦。
sqrt(x)	返回数的平方根。
tan(x)	返回角的正切。
toSource()	返回该对象的源代码。
valueOf()	返回 Math 对象的原始值。

– 向上取整 ceil()

ceil() 方法可对一个数进行向上取整。

语法:

```
Math.ceil(x)
```

参数说明:

参数	描述
x	必需。必须是一个数值。

注意：它返回的是大于或等于 x，并且与 x 最接近的整数。

我们将把 ceil() 方法运用到不同的数字上，代码如下：

```
<script type="text/javascript">
  document.write(Math.ceil(0.8) + "<br />")
  document.write(Math.ceil(6.3) + "<br />")
  document.write(Math.ceil(5) + "<br />")
  document.write(Math.ceil(3.5) + "<br />")
  document.write(Math.ceil(-5.1) + "<br />")
  document.write(Math.ceil(-5.9))
</script>
```

运行结果：

```
1
7
5
4
-5
-5
```

– 向下取整 floor()

floor() 方法可对一个数进行向下取整。

语法：

```
Math.floor(x)
```

参数说明：

参数	描述
x	必需。必须是一个数值。

注意：返回的是小于或等于 x，并且与 x 最接近的整数。

我们将在不同的数字上使用 floor() 方法，代码如下：

```
<script type="text/javascript">
  document.write(Math.floor(0.8)+ "<br>")
  document.write(Math.floor(6.3)+ "<br>")
  document.write(Math.floor(5)+ "<br>")
  document.write(Math.floor(3.5)+ "<br>")
  document.write(Math.floor(-5.1)+ "<br>")
  document.write(Math.floor(-5.9))
</script>
```

运行结果：

```
0
6
5
3
-6
-6
```

– 四舍五入 round()

round() 方法可把一个数字四舍五入为最接近的整数。

语法：

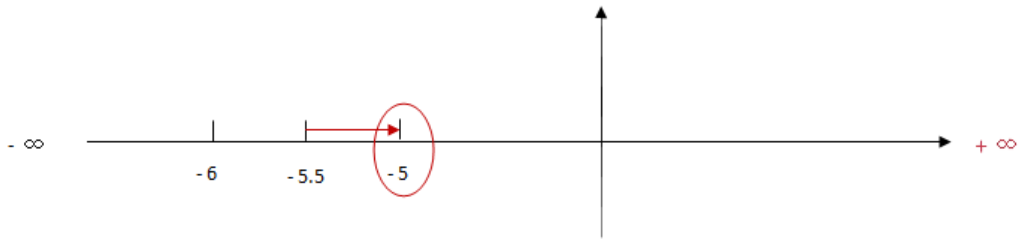
```
Math.round(x)
```

参数说明：

参数	描述
x	必需。必须是数字。

注意：

1. 返回与 x 最接近的整数。
2. 对于 0.5，该方法将进行上舍入。(5.5 将舍入为 6)
3. 如果 x 与两侧整数同等接近，则结果接近 +∞ 方向的数字值。(如 -5.5 将舍入为 -5; -5.52 将舍入为 -6)，如下图：



把不同的数舍入为最接近的整数，代码如下：

```
<script type="text/javascript">
  document.write(Math.round(1.6)+ "<br>");
  document.write(Math.round(2.5)+ "<br>");
  document.write(Math.round(0.49)+ "<br>");
  document.write(Math.round(-6.4)+ "<br>");
  document.write(Math.round(-6.6));
</script>
```

运行结果：

```
2
3
0
-6
-7
```

– 随机数 random()

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

语法：

```
Math.random();
```

注意：返回一个大于或等于 0 但小于 1 的符号为正的数字值。

我们取得介于 0 到 1 之间的一个随机数，代码如下：

```
<script type="text/javascript">
  document.write(Math.random());
</script>
```

运行结果：

```
0.190305486195328
```

注意：因为是随机数，所以每次运行结果不一样，但是 $0 \sim 1$ 的数值。

获得 $0 \sim 10$ 之间的随机数，代码如下：

```
<script type="text/javascript">
  document.write((Math.random())*10);
</script>
```

运行结果：

```
8.72153625893887
```

- 7-17 Array 数组对象

  数组属性：

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

  数组方法：

方法	描述
concat()	连接两个或更多的数组，并返回结果。
join()	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()	删除并返回数组的最后一个元素
push()	向数组的末尾添加一个或更多元素，并返回新的长度。
reverse()	颠倒数组中元素的顺序。
shift()	删除并返回数组的第一个元素
slice()	从某个已有的数组返回选定的元素
sort()	对数组的元素进行排序
splice()	删除元素，并向数组添加新元素。
toSource()	返回该对象的源代码。
toString()	把数组转换为字符串，并返回结果。
toLocaleString()	把数组转换为本地数组，并返回结果。
unshift()	向数组的开头添加一个或更多元素，并返回新的长度。
valueOf()	返回数组对象的原始值

– 数组连接 concat()

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

语法

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

参数说明：

参数	说明
array1	要连接的第一个数组。
...	...
arrayN	第 N 个数组。

注意：该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。

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

```
<script type="text/javascript">
  var mya = new Array(3);
  mya[0] = "1";
  mya[1] = "2";
  mya[2] = "3";
  document.write(mya.concat(4,5)+"<br>");
  document.write(mya);
</script>
```

运行结果：

```
1,2,3,4,5
1,2,3
```

我们创建了三个数组，然后使用 concat() 把它们连接起来，代码如下：

```
<script type="text/javascript">
  var mya1= new Array("hello!")
  var mya2= new Array("I","love");
  var mya3= new Array("JavaScript","!");
  var mya4=mya1.concat(mya2,mya3);
  document.write(mya4);
</script>
```

运行结果：

```
hello!,I,love,JavaScript,!
```

– 指定分隔符连接数组元素 join()

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

语法：

```
arrayObject.join(分隔符)
```

参数说明：

参数	描述
separator	可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

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

```
<script type="text/javascript">
  var myarr = new Array(3);
  myarr[0] = "I";
```

```
  myarr[1] = "love";
  myarr[2] = "JavaScript";
  document.write(myarr.join());
</script>
```

运行结果：

```
I,love,JavaScript
```

我们将使用分隔符来分隔数组中的元素，代码如下：

```
<script type="text/javascript">
  var myarr = new Array(3)
  myarr[0] = "I";
  myarr[1] = "love";
  myarr[2] = "JavaScript";
  document.write(myarr.join("."));
</script>
```

运行结果：

```
I.love.JavaScript
```

– 颠倒数组元素顺序 reverse()

reverse() 方法用于颠倒数组中元素的顺序。

语法：

```
arrayObject.reverse()
```

注意：该方法会改变原来的数组，而不会创建新的数组。

定义数组 myarr 并赋值，然后颠倒其元素的顺序：

```
<script type="text/javascript">
  var myarr = new Array(3)
  myarr[0] = "1"
  myarr[1] = "2"
  myarr[2] = "3"
  document.write(myarr + "<br />")
  document.write(myarr.reverse())
</script>
```

运行结果：

```
1,2,3
3,2,1
```

– 选定元素 slice()

slice() 方法可从已有的数组中返回选定的元素。

语法

```
arrayObject.slice(start,end)
```

参数说明：

参数	描述
start	必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。
end	可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。

1. 返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。
2. 该方法并不会修改数组，而是返回一个子数组。

注意：

1. 可使用负值从数组的尾部选取元素。
2. 如果 end 未被规定，那么 slice() 方法会选取从 start 到数组结尾的所有元素。
3. String.slice() 与 Array.slice() 相似。

我们将创建一个新数组，然后从其中选取的元素，代码如下：

```
<script type="text/javascript">
  var myarr = new Array(1,2,3,4,5,6);
  document.write(myarr + "<br>");
  document.write(myarr.slice(2,4) + "<br>");
  document.write(myarr);
</script>
```

运行结果：

```
1,2,3,4,5,6
3,4
1,2,3,4,5,6
```

– 数组排序 sort()

sort() 方法使数组中的元素按照一定的顺序排列。

语法：

```
arrayObject.sort(方法函数)
```

参数说明：

参数	描述
方法函数	可选。规定排序顺序。必须是函数。

1. 如果不指定，则按 unicode 码顺序排列。
2. 如果指定，则按所指定的排序方法排序。

```
myArray.sort(sortMethod);
```

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

若返回值 <=-1，则表示 A 在排序后的序列中出现在 B 之前。若返回值 >-1 && <1，则表示 A 和 B 具有相同的排序顺序。若返回值 >=1，则表示 A 在排序后的序列中出现在 B 之后。

1. 使用 sort() 将数组进行排序，代码如下：

```
<script type="text/javascript">
  var myarr1 = new Array("Hello","John","love","JavaScript");
  var myarr2 = new Array("80","16","50","6","100","1");
  document.write(myarr1.sort()+"<br>");
  document.write(myarr2.sort());
</script>
```

运行结果：

```
Hello,JavaScript,John,love
1,100,16,50,6,80
```

注意：上面的代码没有按照数值的大小对数字进行排序。

2. 如要实现这一点，就必须使用一个排序函数，代码如下：

```
<script type="text/javascript">
  function sortNum(a,b) {
  return a - b;
 //升序，如降序，把“a - b”该成“b - a”
}
 var myarr = new Array("80","16","50","6","100","1");
  document.write(myarr + "<br>");
  document.write(myarr.sort(sortNum));
</script>
```

运行结果：

```
80,16,50,6,100,1
1,6,16,50,80,100
```

注意：sort(sortNum) 中的 sortNum 是不带括号的。带括号的是把返回值赋值给事件，不带括号的是把函数体所在地址位置赋值给事件。再看 myarr.sort(sortNum)：它的语法定义：arrayObject.sort(方法函数)，里面必须是一个函数，而不是一个返回值或者别的。

8 浏览器对象

9 DOM 对象，控制 HTML 元素

10 编程挑战