简介

使用 `<script>` 标签在HTML网页中插入JavaScript代码。注意， `<script>` 标签要成对出现，并把JavaScript代码写在 `<script></script>` 之间。 `<script type="text/javascript">` 表示在 `<script></script>` 之间的是文本类型(text),javascript是为了告诉浏览器里面的文本是属于JavaScript语言。

JavaScript代码只能写在HTML文件中吗?当然不是，我们可以把HTML文件和JS代码分开,并单独创建一个JavaScript文件(简称JS文件),其文件后缀通常为 `.js` ，然后将JS代码直接写在JS文件中。JS文件不能直接运行，需嵌入到HTML文件中执行，我们需在HTML中添加如下代码，就可将JS文件嵌入HTML文件中。

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

我们可以将JavaScript代码放在html文件中任何位置，但是我们一般放在网页的head或者body部分。 放在 `<head>` 部分 最常用的方式是在页面中head部分放置 `<script>` 元素，浏览器解析head部分就会执行这个代码，然后才解析页面的其余部分。 放在 `<body>` 部分 JavaScript代码在网页读取到该语句的时候就会执行。

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

认识语句和符号

语句

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

注意:

1. `;` 分号要在英文状态下输入，同样，JS中的代码和符号都要在英文状态下输入。
2. 虽然分号 `;` 也可以不写，但我们要养成编程的好习惯，记得在语句末尾写上分号。

注释

单行注释，在注释内容前加符号 `//` 。多行注释以 `/*` 开始，以 `*/` 结束。

变量

定义变量使用关键字 `var` ：`var 变量名`

变量名可以任意取名，但要遵循命名规则:

1. 变量必须使用字母、下划线 `_` 或者美元符 `$` 开始。
2. 然后可以使用任意多个英文字母、数字、下划线 `_` 或者美元符 `$` 组成。
3. **不能使用JavaScript关键词与JavaScript保留字。**

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	

变量要先声明再赋值，如下：

```
var mychar, mynum;
mychar="javascript"; //这里=的作用是给变量赋值，字符串需要用""括起来
var mynum = 6;
```

变量可以重复赋值，如下：

```
var mychar;
mychar="javascript";
mychar="hello";
```

注意:

1. 变量可以是数值、字符串、布尔值等
2. 在JS中区分大小写，如变量**mychar**与**myChar**是不一样的，表示是两个变量。
3. 变量虽然也可以不声明，直接使用，但不规范，需要先声明，后使用。

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）。

操作符

操作符是用于在JavaScript中指定一定动作的符号。

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。

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`

我们都知道，除法、乘法等操作符的优先级比加法和减法高，如果我们要改变运算顺序，需添加括号的方法来改变优先级。

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。

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

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

多重判断（`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}
```

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

代码表示如下:

```
<script type="text/javascript">
  var myscore = 86;
  if(myscore < 60){
      document.write("成绩不及格，加油了！");
  }else if(myscore < 75){
      document.write("成绩良好，不错啊");
  }else if(myscore < 85){
      document.write("成绩很好，很棒");
  }else{
      document.write("成绩优秀，超级棒");
  }
</script>
```

多种选择（`switch`语句)

当有很多种选项的时候，switch比if else使用更方便。

语法:

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

语法说明:

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

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

代码如下:

```
<script type="text/javascript">
var myscore = 6; //myscore变量存储分数,假设为6
switch (myscore) //switch实现判断, case 6匹配
{
 case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 case 5:
   degree = "继续努力！";
   document.write("评语: "+degree+"<br>");
   break;
 case 6:
   degree = "及格，加油!";
   document.write("评语:"+degree+"<br>");
   break;
 case 7:
   degree = "凑合，奋进!";
   document.write("评语: "+degree+"<br>");
   break;
 case 8:
   degree = "很棒，很棒！";
   document.write("评语:"+degree+"<br>");
   break;
 case 9:
 case 10:
   degree = "高手，大牛！";
   document.write("评语: "+degree);
 }
 </script>
```

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

重复重复（`for`循环）

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

for语句结构：

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

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

```
<script type="text/javascript">
var num=1;
for (num=1;num<=6;num++)  //初始化值；循环条件；循环后条件值更新
{   document.write("取出第"+num+"个球<br />");
}
</script>
```

反反复复(while 循环)

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

while语句结构：

```
while(判断条件)
{
    循环语句
 }
```

使用while循环，完成从盒子里取球的动作，每次取一个，共6个球。

```
<script type="text/javascript">
var num=0;  //初始化值
while (num<=6)   //条件判断
{
  document.write("取出第"+num+"个球<br />");
  num=num+1;  //条件值更新
}
</script>
```

来来回回(do...while 循环)

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

do...while语句结构：

```
do
{
    循环语句
 }
while(判断条件)
```

我们试着输出5个数字。

```
<script type="text/javascript">
   num= 1;
   do
   {
     document.write("数值为:" +  num+"<br />");
     num++; //更新条件
   }
   while (num<=5)
</script>
```

退出循环 break

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

格式如下：

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

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

```
<script type="text/javascript">
    var num;
    for(num=1;num<=10;num++)
   {
     if (num==5)
     {
       break;//如果num是5,退出循环。
     }
    document.write("数值:"+num+" <br />");
   }
</script>
```

注:当num=5的时候循环就会结束，不会输出后面循环的内容。

继续循环`continue`

`continue`的作用是仅仅跳过本次循环，而整个循环体继续执行。

语句结构：

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

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

```
<script type="text/javascript">
    var num;
    for(num=1;num<=10;num++)
   {
     if (num==5)
     {
      continue;
     }
    document.write("数值:"+num+"<br />");
   }
</script>
```

数组

我们知道变量用来存储数据，一个变量只能存储一个内容。假设你想存储10个人的姓名或者存储20个人的数学成绩，就需要10个或20个变量来存储，如果需要存储更多数据，那就会变的更麻烦。我们用数组解决问题，一个数组变量可以存放多个数据。

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。

如何创建数组

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。**

数组赋值

我们还可以用简单的方法创建上面的数组和赋值：

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

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

注意：数组存储的数据可以是任何类型（数字、字符、布尔值等）

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。

```
myarray[5]=88; //使用一个新索引，为数组增加一个新元素
```

和C,Java不一样，但是Python居然可以·

数组属性length

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

语法：

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

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

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
```

二维数组

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。

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
   }
 }
```

二维数组的定义方法二*

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

赋值

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

说明: `myarr[0][1]` ，0表示表的行，1表示表的列。

函数

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

如何定义一个函数呢？基本语法如下:

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

说明:

1. function定义函数的关键字。
2. "函数名"你为函数取的名字。
3. "函数代码"替换为完成特定功能的代码。

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

```
function add2(x,y)
{
   sum = x + y;
   return sum; //返回函数值,return后面的值叫做返回值。
}
result = add2(3,4);//语句执行后,result变量中的值为7。
```

1. 没用参数时调用函数记得加 `()`
2. 注意:参数可以多个，根据需要增减参数个数。参数之间用 `,` 隔开。

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>
```

常用互动方法

输出内容 document.write

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

第一种:输出内容用 `""` 括起，直接输出 `""` 号内的内容。

```
<script type="text/javascript">
  document.write("I love JavaScript! "); //内容用""括起来，""里的内容直接输出。
</script>
```

第二种:通过变量，输出内容

```
<script type="text/javascript">
  var mystr="hello world!";
  document.write(mystr);  //直接写变量名，输出变量存储的内容。
</script>
```

第三种:输出多项内容，内容之间用 `+` 号连接。

```
<script type="text/javascript">
  var mystr="hello";
  document.write(mystr+"I love JavaScript"); //多项内容之间用+号连接
</script>
```

第四种:输出HTML标签，并起作用，标签使用 `""` 括起来。

```
<script type="text/javascript">
  var mystr="hello";
document.write(mystr+"<br>");//输出hello后，输出一个换行符
  document.write("JavaScript");
</script>
```

因为浏览器显示机制，对手动敲入的空格，将连续多个空格显示成1个空格。所以我们可以输出html标签 ` ` ，或者使用css样式，在输出是添加 `white-space:pre;` 样式属性，表示空白会被浏览器保留

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

警告 alert 消息对话框

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

语法:

```
alert(字符串或变量);
```

看下面的代码:

```
<script type="text/javascript">
   var mynum = 30;
   alert("hello!");
   alert(mynum);
</script>
```

注:alert弹出消息对话框(包含一个确定按钮)。

结果:按顺序弹出消息框

注意:

1. 在点击对话框**确定**按钮前，不能进行任何其它操作。
2. 消息对话框通常可以用于调试程序。
3. `alert` 输出内容，可以是字符串或变量，与 `document.write` 相似。

确认 `confirm` 消息对话

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

语法:

```
confirm(str);
```

参数说明:

`str` ： 在消息对话框中要显示的文本 返回值: Boolean值

返回值:

当用户点击**确定**按钮时，返回 `true` 当用户点击**取消**按钮时，返回 `false`

注: 通过返回值可以判断用户点击了什么按钮

看下面的代码:

```
<script type="text/javascript">
    var mymessage=confirm("你喜欢JavaScript吗?");
    if(mymessage==true)
    {   document.write("很好,加油!");   }
    else
    {  document.write("JS功能强大，要学习噢!");   }
</script>
```

注: 消息对话框是排它的，即用户在点击对话框按钮前，不能进行任何其它操作。

提问 `prompt` 消息对话框

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

语法:

```
prompt(str1, str2);
```

参数说明:

`str1`：要显示在消息对话框中的文本，不可修改 `str2` ： 文本框中的内容，可以修改

返回值:

1. 点击确定按钮，文本框中的内容将作为函数返回值
2. 点击取消按钮，将返回null

看看下面代码:

```
var myname=prompt("请输入你的姓名:");
if(myname!=null)
  {   alert("你好"+myname); }
else
  {  alert("你好 my friend.");  }
```

注:在用户点击对话框的按钮前，不能进行任何其它操作。

打开新窗口 window.open

`open()` 方法可以查找一个已经存在或者新建的浏览器窗口。

语法:

```
window.open([URL], [窗口名称], [参数字符串])
```

参数说明:

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

1. 该名称由字母、数字和下划线字符组成。
2. `_top` 、 `_blank` 、 `_self` 具有特殊意义的名称。 `_blank` ：在新窗口显示目标网页 `_self` ：在当前窗口显示目标网页 `_top` ：框架网页中在上部窗口中显示目标网页
3. 相同 name 的窗口只能创建一个，要想创建多个窗口则 name 不能相同。
4. name 不能包含有空格。

`参数字符串` ：可选参数，设置窗口参数，各参数用逗号隔开。

参数表:

参数	值	说明
top	Number	窗口顶部离开屏幕顶部的像素数
left	Number	窗口左端离开屏幕左端的像素数
width	Number	窗口的宽度
height	Number	窗口的高度
menubar	yes,no	窗口有没有菜单
toolbar	yes,no	窗口有没有工具条
scrollbars	yes,no	窗口有没有滚动条
status	yes,no	窗口有没有状态栏

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

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

注意：运行结果考虑浏览器兼容问题。

关闭窗口 window.close

`close()` 关闭窗口

用法:

```
window.close();   //关闭本窗口
```

或

```
<窗口对象>.close();   //关闭指定的窗口
```

例如:关闭新建的窗口。

```
<script type="text/javascript">
   var mywin=window.open('http://www.imooc.com'); //将新打的窗口对象，存储在变量mywin中
   mywin.close();
</script>
```

注意:上面代码在打开新窗口的同时，关闭该窗口，看不到被打开的窗口。

DOM

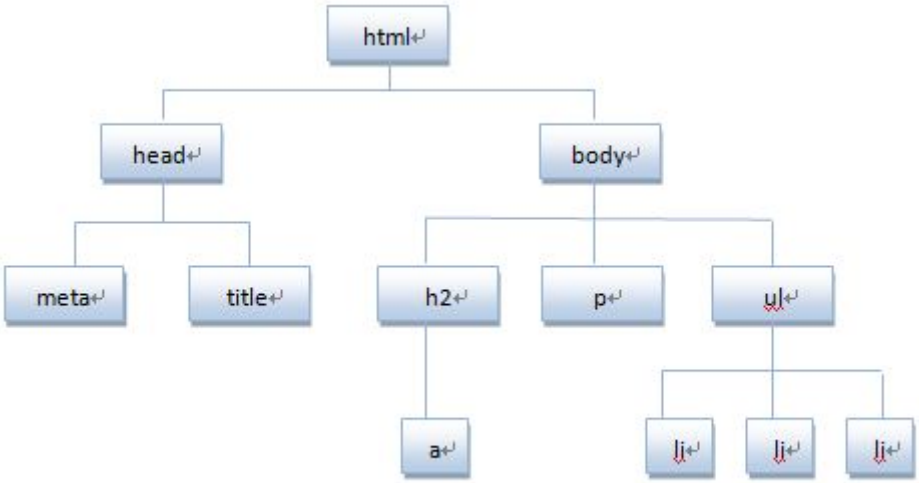
节点简介

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

先来看看下面代码:

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>DOM</title>
</head>
<body>
    <h2><a href="http://www.imooc.com">javascript DOM</a></h2>
    <p>对HTML元素进行操作，可添加、改变或移除CSS样式等</p>
    <ul>
        <li>JavaScript</li>
        <li>DOM</li>
        <li>CSS</li>
    </ul>
</body>
</html>
```

将HTML代码分解为DOM节点层次图:



HTML文档可以说由节点构成的集合，三种常见的DOM节点:

1. **元素节点**：上图中 `<html>` 、 `<body>` 、 `<p>` 等都是元素节点，即标签。
2. **文本节点**:向用户展示的内容，如< `li>...</li>` 中的JavaScript、DOM、CSS等文本。
3. **属性节点**:元素属性，如 `<a>` 标签的链接属性 `href="http://www.imooc.com"` 。

看下面代码:

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

通过ID获取元素

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

语法:

```
1    document.getElementById("id")
```

看看下面代码:

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>获取元素</title>
  <script type="text/javascript">
    var mye = document.getElementById("con");//获取元素存储在变量mye中
    document.write(mye);//输出变量mye
  </script>
</head>

<body>
    <h3>Hello</h3>
    <p id="con">I love JavaScript</p>
</body>
</html>
```

结果: `nul` l或 `[object HTMLParagraphElement]`

注:获取的元素是一个对象，如想对元素进行操作，我们要通过它的属性或方法。

`innerHTML` 属性

`innerHTML` 属性用于获取或替换 HTML 元素的内容。

语法:

```
Object.innerHTML
```

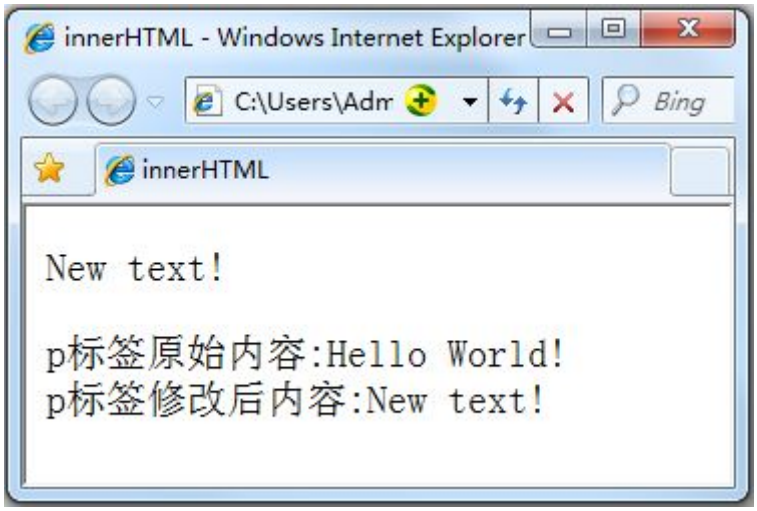
注意:

1. **Object是获取的元素对象**，如通过 `document.getElementById("ID")` 获取的元素。
2. 注意书写，innerHTML区分大小写。

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

```
<!DOCTYPE HTML>
<html>
<head>
<title>innerHTML</title>
</head>
<body>
   <p id="con">Hello World!</p>
   <script>
       var mycon= document.getElementById("con");
       document.write("p标签原始内容:"+ mycon.innerHTML+"<br>");
        //输入元素内容
       mycon.innerHTML ="New text!"; //修改p元素内容
       document.write("p标签修改后内容:"+ mycon.innerHTML);
   </script>
</body>
</html>
```

结果:



改变 HTML 样式

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

语法:

```
Object.style.property=new style;
```

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

基本属性表（property）：

属性	描述
backgroundColor	设置元素的背景颜色
height	设置元素的高度
width	设置元素的宽度
color	设置文本的颜色
font	在一行设置所有的字体属性
fontFamily	设置元素的字体系列。
fontSize	设置元素的字体大小。

注意:该表只是一小部分CSS样式属性，其它样式也可以通过该方法设置和修改。

看看下面的代码:

改变 `<p>` 元素的样式，将颜色改为红色，字号改为20,背景颜色改为蓝:

```
<p id="pcon">Hello World!</p>
<script>
   var mychar = document.getElementById("pcon");
   mychar.style.color="red";
   mychar.style.fontSize="20";
   mychar.style.backgroundColor ="blue"; //注意加""
</script>
```

显示和隐藏（`display` 属性）

网页中经常会看到显示和隐藏的效果，可通过 `display` 属性来设置。

语法：

```
Object.style.display = value
```

value取值:

值	描述
none	此元素不会被显示(即隐藏)
block	此元素将显示为块级元素(即显示)

看看下面代码:

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>display</title>
    <script type="text/javascript">
        function hidetext()
        {
            document.getElementById("con").style.display="none";
        }
        function showtext()
        {
            document.getElementById("con").style.display="block";
        }
    </script>
</head>
<body>
    <h1>JavaScript</h1>
    <p id="con">
做为一个Web开发师来说，如果你想提供漂亮的网页、令用户满意的上网体验，JavaScript是必不可少的工具。</p>
    <form>
       <input type="button" onclick="hidetext()" value="不显示段落内容" />
       <input type="button" onclick="showtext()" value="显示段落内容" />
    </form>
</body>
</html>
```

控制类名（`className`属性）

`className`属性设置或返回元素的`class`属性。

语法:

```
object.className = classname
```

作用:

1. 获取元素的class 属性
2. 为网页内的某个元素指定一个css样式来更改该元素的外观

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

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>className属性</title>
<style type="text/css">
    input{
        font-size:10px;
        }
    .one{
        width:200px;
        background-color:#CCC;}
    .two{
        font-size:18px;
        color:#F00;
        }
</style>
</head>
<body>
    <p id="con" class="one">JavaScript</p>
    <form>
        <input type="button" value="点击更改" onclick="modifyclass()"/>
    </form>
    <script type="text/javascript">
        var mychar= document.getElementById('con');
        document.write("p元素Class值为: " + mychar.className+"<br>");
        // 输出p元素Class属性
        function modifyclass(){
            mychar.className="two"; //改变className
        }
    </script>
</body>
</html>
```

FAQ

Are braces necessary in one-line statements in JavaScript?

It is highly recommended that you always use braces because if you (or someone else) ever expands the statement it will be required.

This same practice follows in all C syntax style languages with bracing. C, C++, Java, even PHP all support one line statement without braces. You have to realize that you are only saving two characters and with some people's bracing styles you aren't even saving a line. I prefer a full brace style (like follows) so it tends to be a bit longer. The tradeoff is met very well with the fact you have extremely clear code readability.

Does the position of CSS properties order mater or not?

The order of the properties does not matter (in a technical sense) unless they:

- Are the same property (including vendor prefixed versions)
- One property is a shorthand property that includes the other

BTW I think if the effects of the properties are not overlapped, it's not matter.

Are custom elements valid HTML5?

Yes, you can. But check **stackflow**

Note

在js中，arr索引是不能用-1的，但是string中可以用-1索引。+=亲测有效。