# Web程序设计

## 第5章 JavaScript语言基础


NCRE

# 第5章 JavaScript语言基础

# 5.1 JavaScript的作用和特点

- JavaScript语言是一门解释性的脚本语言
- 嵌入HTML文档中的JavaScript源代码是作为HTML文档的一部分而存在的
- 在用户使用任何一种支持JavaScript语言的浏览器来浏览具有JavaScript源代码的HTML网页时，浏览器本身就可以对该HTML网页进行分析和识别，最终解释并执行用JavaScript语言编写的源代码。

# 5.1 JavaScript的作用和特点

- JavaScript可以分为三个部分：核心、客户机端和服务器端。
- 核心是语言的内核，包含操作符、表达式、语句和子程序。
- 在客户机端应用的JavaScript是一组对象的集合，利用这些对象可以对浏览器和用户交互进行控制。比如，利用JavaScript、HTML文档可以对单击鼠标和键盘操作等用户输入信息作出反应。
- 应用于服务器端的JavaScript也是一组对象的集合，这些对象可以应用于Web服务器编程，比如，支持与数据库管理系统之间的通信。

# 5.1 JavaScript的作用和特点

**JavaScript的特点：**

●分布式运算。在**JavaScript**脚本描述语言中，允许多种任务仅在客户机上就可以完成而不需要网络环境和服务器参与，从而实现分布式运算。

●安全可靠。**JavaScript**是构筑在**Java**语言和控制结构上的脚本描述语言，不允许访问本地硬盘，不允许将数据存放到服务器中，只能通过浏览器实现信息浏览和动态交互。

●容易移植。在提交数据前由客户机的**JavaScript**语言实现自动验证处理，这些操作是与具体计算机系统无关的。

●使用浏览器对象。在**JavaScript**脚本描述语言中，用户可以非常方便地操纵各种浏览器对象，并对浏览器的外观、状态、运行模式进行控制。

# 5.2 JavaScript语法

## 5.2.1 常量

（1）整型常量。以0开头的是八进制数，例如065。以0x开头的是十六进制数，例如0xff。其余的都是十进制。

（2）浮点常量。如12.8、1.2e-7。

（3）布尔常量。有true和false。

（4）字符串常量。如"你好"、"123"。

（5）转义字符。如' \n'代表换行，' \t'代表Tab符。

（6）Undefine。表示没被定义的值，不能用于计算。

（7）Null。代表空值，不能用于计算。

# 5.2 JavaScript语法

## 5.2.2 变量

JavaScript中的变量声明用关键字var引导，变量的类型取决于它的声明方式。

① 声明原始类型的变量：

```
var 变量名 = 值;
```

② 声明引用类型的变量：

```
var 变量名 = new 类型名(初值);
```

③ JavaScript的变量是弱类型的，变量的类型主要取决于它的值，用户可以随时更改它的类型。如：

```
var i;
i = 20;
i = "red";
```

# 5.2 JavaScript语法

## 5.2.3 类型

JavaScript的数据类型可分为两大类：原始类型、引用类型。原始类型变量的值存放在栈中，访问变量时可直接访问到变量的值。引用类型变量在栈中存放的是指针，其值另外存放在堆中，需要通过指针才能访问。

（1）原始类型：包括Undefined、Null、Boolean、Number、String五种。

（2）引用类型：所有对象都是引用类型，包括Boolean、Number、String、Array、Date、Math等。其中，Boolean、Number、String既可以是原始类型，也可以是引用类型。

引用类型的变量除了可以访问变量的值以外，还提供了若干属性和方法

# 5.2 JavaScript语法

## 5.2.4 运算符

JavaScript中的运算符和C语言基本相同，优先级也相同。

●算术运算符：+（加）、-（减）)、*（乘）、/（除）、%（取模）、++（增量）、--（减量）、-（取负）。

●逻辑运算符：&&（与）、||（或）、!（非）。

●比较运算符：>（大于）、>=（大于等于）、<（小于）、<=（小于等于）、==（等于）、!=（不等于）、===（严格相等）、!==（不严格相等）。注：“相等”只要求值相等，“严格相等”要求值相等且类型相同。如：1=="1" 是 true，而 1==="1" 是 false。

●位运算符：&（位与）、|（位或）、^（位异或）、~（按位取反）、<<（左移）、>>（右移）、>>>（填0右移）。

# 5.2 JavaScript语法

## 5.2.4 运算符

●赋值运算符：=（赋值）、op=（运算赋值）。注：运算赋值中的运算符可以是算术运算符或位运算符。如：“x+=10”表示“x=x+10”。

●条件运算符：条件?表达式1: 表达式2。注：若条件为true，值为表达式1的值，否则为表达式2的值。

●字符串连接运算符：+、+=。如："every"+"one"的结果为"everyone"。注：连接的项目中可以包括字符串、常量、变量，不需要都是字符串，只要项目中有一个是字符串就会按字符串进行连接。如："ex"+20+"b"的结果为"ex20b"。

●new运算符：用于创建一个对象。如：new Array() 表示生成一个数组对象。

●delete运算符：用于删除一个对象。

# 5.2 JavaScript语法

## 5.2.5 语句

（1）表达式语句：x=1; x++;

（2）条件语句：if … else …

（3）多路分支语句：switch … case …

```
switch( a )
{
case 1: x = "A"; break;
case 2: x = "B"; break;
case 3: x = "C"; break;
default: x = "D";
}
```

# 5.2 JavaScript语法

## 5.2.5 语句

（4）循环语句：while

```
while( x<5 ) {  y += x; x++; }
```

（5）循环语句：do … while

```
do{   y += x; x++;
}while( x<5 );
```

（6）循环语句：for

```
for( i=1; i<=10; i++ ){  x += 2;}
```

（7）退出循环：break、continue

break语句可退出循环语句或switch语句。

continue语句可结束本次循环，并开始下一次循环。

# 5.3 JavaScript函数

- 函数定义包含了函数的标题与一组复合语句，这组复合语句称为函数的主体，用于描述函数的操作。函数标题包含function、函数的名称和一对括号，括号中包含了函数的参数。

JavaScript的函数用关键字function定义。定义格式为：

function 函数名(参数表)

{ 函数体}

函数是通过函数调用来执行的：

函数名(实际参数)

# 5.3 JavaScript函数

- 函数示例

```
var a, b;
function switch()
{  var t;
  t = a;  a = b;  b = t; }
```

本例是一个交换两个变量值的函数，由于一个函数只能返回一个值，所以只能通过全局变量获得交换的结果。

说明：**JavaScript**不支持行内变量定义，所以函数内的所有变量最好都在函数开始处声明，这样才能保证它们是局部变量。如果在语句中应用了一个未声明的变量，它会自动成为全局变量，有可能影响其他函数的执行。

# 5.4 JavaScript的使用方法

可以将<script>.....</script>标识放入<head>...</head> 或<body></body>之间。

```
<html>
<body>
<script type=”text/Javascript”>
  document.write(“Hello, world”);
  document.write(“<br><p>”);
  document.write(“第一个Javascript程序”);
</script>
</body>
</html>
```

# 5.4 JavaScript的使用方法

- 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**。
- 外部**.js**文件中不能有**<script> </script>**标识。

```
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

# 5.5 JavaScript内置对象

内置对象为编程提供几种最常用的功能。JavaScript内置对象有以下几种。

- String对象：处理所有的字符串操作。
- Math对象：处理所有的数学运算。
- Date对象：处理日期和时间的存储、转化和表达。
- Array对象：提供一个数组的模型，存储大量有序的数据。
- Global对象：提供JavaScript初始化时的特殊对象。

内置对象都有自己的方法和属性，访问方法如下：

对象名.属性名称

对象名.方法名称(参数表)

# 5.5 JavaScript内置对象

## 5.5.1 String对象

String对象提供了对字符串的支持。使用如下的方式可以创建一个String对象：

var str= new String ("test"); 或者： var str=”test”;

String 对象的属性length表示字符串中的字符个数。

String对字符串的处理提供了丰富的方法。如：

```
ar txt="Hello world!"
document.write(txt.toUpperCase())
document.write(txt.length)
```

上面的代码输出为：

HELLO WORLD!

12

# 5.5 JavaScript内置对象

## 5.5.2 Math对象

❖ Math对象提供了一组数学常数的定义和数学函数，使用Math对象可以在JavaScript中进行常用的数学运算。

```
document.write(Math.round(4.8))
```

输出为：5

```
document.write(Math.floor(Math.random()*11))
```

输出为：一个介于 0 和 10 之间的随机数

# 5.5 JavaScript内置对象

## 5.5.3 Date对象

Data对象提供了与日期、时间相关的操作功能，Date对象具有多种构造函数，常用的有：

new Date()

new Date(datestring)

new Date(year,month)

new Date(year,month,day)

如：创建一个代表当前时间的date对象

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

创建Date对象，指定年、月、日、小时、分、秒、毫秒

```
var d=new Date(2006,5,9,7,40,0,0);
```

# 5.5 JavaScript内置对象

## 5.5.4 Array对象

Array对象是JavaScript的内置对象，它提供了对数组的支持，其作用是使用一个变量名来存取一组数据。通过以下几种方法都可以创建一个Array对象：

```
var arr =new Array( );  //创建一个空数组
var arr=[ ];            //创建一个空数组
var arr=new Array[10]; //创建一个长度为10的数组
var arr=new Array[1,2,3];//创建数组，其元素为1、2、3
```

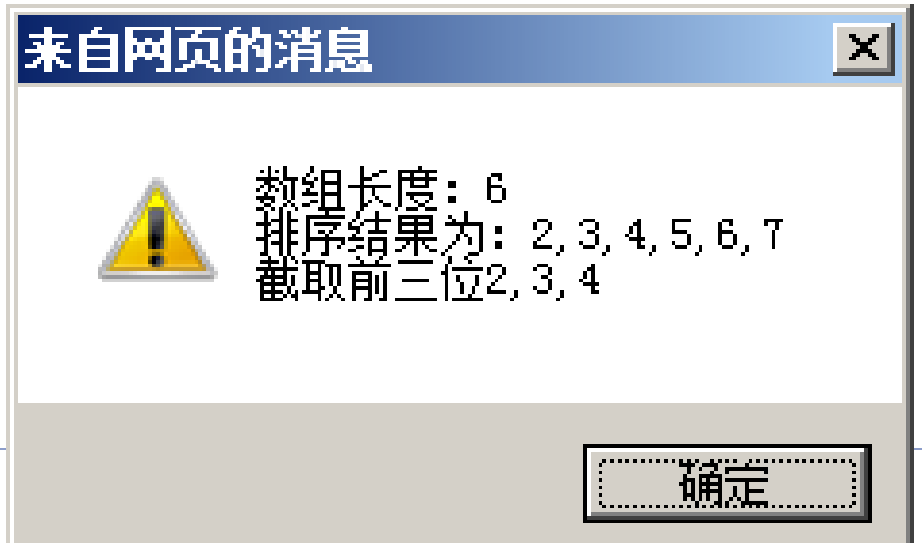
常用Array对象方法包括：concat（array1,arrayn)，join(string),pop(), push(value),reverse()，sort(compare Function), toString()等。

# 5.5 JavaScript内置对象

## 5.5.4 Array对象

数组是通过数组名以及索引号码来访问数组中的元素，数组元素的索引号是从0开始的。如：

```
<script type="text/JavaScript" >
var arr=[5,6,7,2,3,4];
alert("数组长度："+arr.length+"\n"+"排序结果为："+ arr.sort()+"\n"+"截取前三位"+ arr.slice(0,3));
</script>
```



# 5.5 JavaScript内置对象

## 5.5.5 Global对象

❖ Global对象是一种由JavaScript脚本引擎在初始化时创建的特殊对象，在不同的宿主环境下有不同的含义，在浏览器中，Global对象就是浏览器对象模型中的window对象。使用Global对象的属性和方法无需引用Global，这是因为Global对象实际上是一个概念，而非对象的实际名称。

❖ Global对象的属性

- Infinity 指定一个正负无穷大的数值。
- NaN 指定一个“非数字”值。
- undefined 指定一个未被赋值的变量。

# 5.5 JavaScript内置对象

## 5.5.5 Global对象

Global对象的部分方法：

- decodeURI() 为加密的URI进行解码。
- decodeURIComponent() 为加密的URI组件解码。
- encodeURI() 将字符串加密为URI。
- encodeURIComponent()将字符串加密为URI组件。
- escape(string) 加密一个字符串。
- unescape() 使用escape()对一个字符串进行解码。
- eval(string) 判断一个字符串并将其以脚本代码的形式执行。
- isFinite(number)检测一个值是否为一个有限数字，返回true或false。

# 5.5 JavaScript内置对象

## 5.5.5 Global对象

Global对象的部分方法：

- isNaN(string) 检测一个字符串是否是非数字值。
- number(object) 将一个对象的值转换为一个数字。
- parseFloat(string) 将一个字符串解析为一个浮点数字。
- parseInt(string) 将一个字符串解析为一个整数。不是四舍五入操作，而是切尾。
- string(object) 将一个对象值转换为一个字符串。

JavaScript的内置对象还有：对象（object）对象、正则表达式（RegExp）对象、数字（number）对象、函数（function）对象、布尔（Boolean）对象、错误（Error）对象。

# 5.6浏览器对象模型BOM

- 在JavaScript中对象之间并不是独立存在的，对象与对象之间有着层次关系。如document对象是window对象的子对象，window对象是document对象的父对象等。
- 浏览器对象模型（Browser Object Model，BOM）就是用于描述这种对象与对象之间层次关系的模型，该对象模型提供了独立于内容的、可以与浏览器窗口进行互动的对象结构。
- BOM由多个对象组成，其中代表浏览器窗口的window对象是BOM的顶层对象，其他对象都是该对象的子对象。
- BOM提供了一组以window为核心的对象，实现了对浏览器窗口的访问控制。

# 5.6浏览器对象模型BOM

BOM模型中的对象及其关系：

- window对象表示浏览器中打开的窗口。
- document对象表示浏览器中加载页面的文档对象。
- location对象包含了浏览器当前的URL信息。
- navigation对象包含了浏览器本身的信息。
- screen对象包含了客户机屏幕及渲染能力的信息。
- history对象包含了浏览器访问网页的历史信息。

其他的5个对象都是window对象的属性。

# 5.6浏览器对象模型BOM

## 5.6.1 window对象

- window对象表示浏览器中打开的窗口，提供关于窗口状态的信息。可以用window对象访问窗口中绘制的文档、窗口中发生的事件和影响窗口的浏览器特性。
- 浏览器在打开HTML文档时生成window对象。如果文档定义一组帧，则浏览器对原文档生成一个window对象，并对每个帧生成window对象，这些对象是原父窗口的子窗口。
- 也可以通过用showModalDialog方法生成新窗口，从而生成新的window对象。
- window对象主要提供5类功能：（1）调整窗口大小（2）打开新窗口（3）系统对话框（4）状态栏控制（5）定时操作

# 5.6浏览器对象模型BOM

## 5.6.1 window对象

（1）调整窗口的大小和位置

在JavaScript中可以使用window.moveBy、window.moveTo、window.resizeBy和window.resizeTo这4个方法调整窗口的大小和位置。

●window.moveBy(20, 20)：将浏览器左移20px，下移20px

●window.moveTo(20,20)：将浏览器窗口移动到（20,20）的位置上

●window.resizeBy(20,20)：将浏览器窗口的宽度和高度分别增大20px

●window.resizeTo(20,20)：将高度和宽度分别设置为20px。

# 5.6浏览器对象模型BOM

## 5.6.1 window对象

（2） 打开新窗口：window.open，用法如下：

window.open(url,target,options);

其中，

url为打开的浏览器窗口要加载的URL;

target为新打开的浏览器窗口的定位目标或者名称；

options为新打开的窗口的规格。options参数可能的选项包括：

- Height 窗口的高度。
- Width 窗口的宽度。
- Left 窗口左边缘的位置。
- Right 窗口右边缘的位置。
- Top 窗口上边缘的位置。
- Fullscreen 是否是全屏。
- Menubar 是否显示菜单栏
- Toolbar 是否显示工具

# 5.6浏览器对象模型BOM

## 5.6.1 window对象

（3） 系统对话框

当某些事件发生时需要通过系统对话框向用户提供信息，这类方法包括window.alert、window.confirm、window.prompt。

- window.alert该方法将显示消息提示框。
- window.confirm该方法将显示一个确认对话框。
- window.prompt该方法将显示一个消息对话框

示例：

```
var num= window.prompt("请输入一个数字"，"0");
if(num==String(Number(num)))window.alert("您输入的是："+num);
else window.alert("请输入数字");
```

# 5.6浏览器对象模型BOM

## 5.6.1 window对象

（4）状态栏控制

浏览器状态栏的显示信息可以通过window.status属性直接修改。

但是，一般来说不建议对其进行修改，因为用户通常习惯于通过状态栏观察网页加载的状态，修改状态栏可能会对网站使用者造成困惑。

# 5.6浏览器对象模型BOM

## 5.6.1 window对象

（5）定时操作

定时操作通常有两种使用目的，一是周期性地执行脚本，例如在页面上显示时钟，需要每隔一秒更新一次，二是将某个操作延迟一段时间执行，例如某个特别耗时的操作。定时操作的函数有4个：

●window.setInterval该函数用于设置定时器，每隔一段时间执行指定代码。

●window.clearInterval该函数用于清除setInterval函数设置的定时器。

●window.setTimeout该函数用于设置定时器，在一段时间之后执行指定代码，该定时器只执行一次。

●window.clearTimeout该方法用于清除setTimeout函数设置的定时器。

# 5.6浏览器对象模型BOM

## 5.6.2 document对象

document对象实际上是window的属性，从BOM角度看，document对象由一系列集合构成，这些集合可以访问文档的各个部分，并提供页面自身的信息。

document对象的方法都和文字的输出有关，它们是document.open/close、document.write/writeln。示例：

```
var win=window.open();
win.document.open();
win.document.writeln("some text……");
 //写入一些文字
win.document.close();
 //关闭文档，显示写入的文字
```

# 5.6浏览器对象模型BOM

## 5.6.2 document对象

通用属性：

- document.bgColor该属性为页面的背景色。
- document.fgColor该属性为页面的前景色。
- document.linkColor该属性为页面文档中链接的颜色。
- document.vlinkColor该属性为页面中访问过的链接颜色。
- document.alinkColor该属性为页面中激活链接的颜色。
- document.domain 该属性为文档所在的域名。
- document.URL 该属性为当前页面的URL。
- document.title 该属性为当前页面的标题。
- document.cookie该属性用于设置或者读取cookie的值。

# 5.6浏览器对象模型BOM

## 5.6.3 location对象

location对象中包含了当前窗口的url信息，它的对象及详细说明如下：

- location.hash 返回href后面的字符串。
- location.host提供url的name:port部分。
- location.hostname 提供url的hostname部分。
- location.href 提供整个url。
- location.pathname 提供url中第三个斜杠后面的文件名。
- location.port 返回url端口号。
- location.protocol 返回表示url访问方法的首字母子串。
- location.search 提供完整url后面的查询字符串。

# 5.6浏览器对象模型BOM

## 5.6.4 navigator对象

navigator对象保存浏览器厂家、版本和功能的信息，共有5个属性，分别如下：

- navigator.appCodeName提供浏览器的代码名。
- navigator.appName提供浏览器的产品名。
- navigator.appVersion提供浏览器的版本号。
- navigator.cookieEnabled表示浏览器是否允许客户机端cookie。
- navigator.userAgent作为http协议的一部分发送的浏览器名。

# 5.6浏览器对象模型BOM

## 5.6.5 screen对象

screen对象的属性保存最终用户的屏幕分辨率和屏幕绘制文档的功能的信息。这个信息在设置浏览器窗口的特征时很有用。下面列出了screen对象的属性。

- screen.colorDepth返回用户系统支持的最大颜色个数信息。
- screen.height提供用户屏幕的总高度。
- screen.pixelDepth提供系统每个像素占用的位数。
- screen.updateInterval保持用户机器上屏幕更新的间隔。
- screen.width提供用户屏幕的总宽度。

# 5.6浏览器对象模型BOM

## 5.6.6 history对象

history对象保存当前对话中用户访问的url信息。它的length属性提供浏览器历史清单中的项目个数。它可以用来在浏览的历史清单中移动。下面列出了history对象的属性：

- history.length 表示浏览历史记录的总数。
- history.go(index) 表示从浏览历史中加载url。index参数是加载url的相对位置，index为负数时表示当前地址之前的浏览记录，index为正数时表示当前地址之后的浏览记录。
- history.forward 表示从浏览历史中加载下一个url，相当于history.go(1)。
- history.back表示从浏览历史中加载上一个url，相当于history.go(-1)

# 5.7 JavaScript 事件及处理

- HTML4定义了一组事件，浏览器已经实现了这些事件。
- 利用JavaScript 使我们可以对这些事件进行处理，创建动态页面。
- 网页中的每个元素都可以产生某些可以触发 JavaScript 函数的事件。比方说，我们可以在用户点击某按钮时产生一个 onClick 事件来触发某个函数。
- 事件在 HTML 页面中定义。

# 5.7 JavaScript 事件及处理

表 5-1： 常用事件属性及含义

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

# 5.7 JavaScript 事件及处理

- HTML4定义了一组事件，浏览器已经实现了这些事件。
- 利用JavaScript 使我们可以对这些事件进行处理，创建动态页面。
- 网页中的每个元素都可以产生某些可以触发 JavaScript 函数的事件。比方说，我们可以在用户点击某按钮时产生一个 onClick 事件来触发某个函数。
- 事件在 HTML 页面中定义。

# 5.7 JavaScript 事件及处理

## 5.7.1 事件处理程序的引入

- 为某个标签元素对象绑定事件响应函数：
  - 静态引入：事件处理程序的句柄名以属性名的形式在HTML标签实体对象中出现，而对应的属性值为事件响应的代码（事件响应的函数名），从而实现将HTML标签元素对象与JavaScript事件响应的函数名相对应，即事件绑定。
  - 动态引入：将事件响应的函数名直接赋值给HTML页面中某个标签元素的事件属性。

# 5.7 JavaScript 事件及处理

## 5.7.1 事件处理程序的引入

静态引入方式为按钮添加onclick事件：

```
<form name=form1>
Field1: <input type="text" name="field1" value="Hello World!"><br />
Field2: <input type="text" name="field2">
<br /><br />
Click the button below to copy the content of Field1 to Field2. <br />
<input type=”button” value=”copytext” onclick="document.form1.field2.value=document.form1.field1.value">
</form1>
```

# 5.7 JavaScript 事件及处理

## 5.7.2 常用事件

（1）鼠标单击事件句柄 onclick

当操作者的鼠标指向某个HTML标签元素对象并单击鼠标时，将触发onclick事件，此时的onclick事件所对应的事件处理程序或代码将被调用执行。

onclick事件主要出现在按钮、复选框、单选按钮、取消按钮和提交按钮等类型的标签元素对象中。

下面代码用于单击图像后弹出一个对话框：

```
<img src=”myimg.jpg” onclick=”window.alert(‘图像被点击’);” >
```

## 5.7.2 常用事件

（2）MouseDown事件

当按下鼠标的某一个键时发生，在这个事件发生后，Javascript自动调用MouseDowm句柄。在Javascript中，如果一个事件处理函数返回false，就终止事件的继续处理，如果MouseDown事件处理函数返回false，与鼠标操作有关的其它操作如拖放、激活超链接等都会无效。

示例代码：

```
Function whichMouseButton(event){……}
<body onmousedown="whichMouseButton(event)">
<p>可以单击鼠标了</p>
```

# 5.7 JavaScript 事件及处理

## 5.7.2 常用事件

（3）MouseUp事件

释放鼠标时发生该事件，其它同MouseDown事件。

（4）MouseOver事件

当光标移动到一个对象上面时发生该事件，JavaScript自动调用onMouseOver句柄。示例代码：

```
<a href=http://www.163.com onMouseOver
= ”window.status=’你好吗’;
return true”>请点击</a>
```

（5）MouseOut事件

当光标离开一个对象时发生该事件，这个事件适用于区域、层及超链接对象。使用方法同MouseOver事件。

## 5.7.2 常用事件

（6） 文本输入框的内容变化事件句柄onchange

当操作者在单行或者多行文本输入框中输入结束并单击文本框以外的区域，使文本输入框失去焦点时触发该事件，在下拉选择框中选择某个选项时也会触发该事件。

（7） 选中事件句柄onselect

当操作者在单行或者多行文本输入框中对其中的字符进行选择时触发该事件，改变了下拉列表中当前的列表选项时也会触发该事件。

（8） 获得焦点句柄onfocus和失去焦点句柄onblur

当用户单击选中了单行或者多行文本输入框、下拉选择框中的选择项目时，将触发onfocus事件；当单行或者多行文本输入框、下拉选择框对象不再被选中而退到后台时，将触发onblur事件。

## 5.7.2 常用事件

（9）Select事件

选定文本输入框或文本输入区域的一段文字后，发生该事件。在select事件发生后，由JavaScript自动调用OnSelect句柄。

（10） 页面文档的载入事件句柄onload和卸载事件句柄onunload

当浏览器加载完某个页面文档中的各个标签内容（也包括各个图像文件）时将触发onload事件。

当Web页面退出时（如果操作者关闭浏览器或者在浏览器窗口中加载另一个页面文档时）触发onunload事件。载入事件onload和卸载事件onunload都只适用于<body>和<frameset>标签。

## 5.7.2 常用事件

**（11） 图像事件（image events）**

**图像事件只能应用于<img>标签元素中，并且只提供了一个onabort事件，当某个图像文件被加载时触发该事件。**

**（12）键盘事件（keyboard events）**

- 键盘事件主要有onkeydown（当键盘中的某个按键被按下时触发）
- onkeypress（当键盘中的某个按键被按下后又松开时触发）
- onkeyup（当键盘中的某个按键被松开时触发）。

# 5.7 JavaScript 事件及处理

## 5.7.3 JavaScript表单处理

- 表单处理是建立动态网页中最重要的部分， Form对象提供一个让用户输入文字或选择的功能，例如单选按钮、复选框、选择列表等。

- 表单由<form>标记组构成。JavaScript为每一个表单建立一个表单对象，并可以将用户提供的信息送至服务器进行处理，或在JavaScript脚本中编写程序对数据进行处理。

- 当用户单击表单中的“提交”按钮来提交表单中的各个请求时，就会触发表单的onsubmit事件，将表单中的数据打包发送给目标程序进行处理。

# 5.7 JavaScript 事件及处理

## 5.7.3 JavaScript表单处理

示例：

```
<script type="text/JavaScript" >
function test(){
    var username=document.form1.username.value;
    var password=document.form1.password.value;
    alert("用户名："+username+"密码:"+password);}
</script></head>
<body><form name="form1" onsubmit="test()">
username:<input type="text" name="username" /><br>
password:<input type="password"
name="password"/><br>
<input type="submit" value="提交" />
```

# 5.7 JavaScript 事件及处理

## 5.7.3 JavaScript表单处理

❖ 表单中的基本元素包括按钮、单选按钮、复选框、提交按钮、重置按钮及文本框等。在JavaScript中要访问这些基本元素，可以通过对应的表单元素名来实现，并访问这些元素的属性或方法。

（1）text单行单列输入元素和Textarea多行多列输入框，

其属性有:

**name:** 设定提交信息时的信息名称，对应文档中的**name**。

**Value：** 用以设定出现在窗口中对应文档中**value**的信息

**Defaultvalue:** 设定默认值

对应的主要事件有:

**OnFocus, OnBlur,onselect**和**onchange**。

## 5.7.3 JavaScript表单处理

（2）Select选择元素

其属性有：

**name:** 设定提交信息时的信息名称，对应文档中的**name**。

**Value：** 用以设定出现在窗口中对应文档中**value**的信息

**length:** 对应该**select**中的选项数目

**options:** 组成多个选项的数组

**selectIndex:** 指明一个选项

**text:** 选项对应的文字

**selected：** 指明该项是否被选中

**index:** 指明当前选项的位置

**defaultselected:**默认选项

对应的主要事件有：**onBlur, onFocus, onChange.**

# 5.7 JavaScript 事件及处理

## 5.7.3 JavaScript表单处理

（3） checkbox复选框

其属性有：

name: 设定提交信息时的信息名称，对应文档中的name。

Value： 用以设定出现在窗口中对应文档中value的信息

checked: 指明该框的选择状态为true或false

对应的主要事件有：onclick，如：

```
<form name="myForm">
    <input type="checkbox" name="myCheck" value="My Check Box"> check Me</form>
    <a href="#" onClick="window.alert(document.myForm.myCheck.checked?'yes':'No');">Am I Checked?</a>
```

## 5.7.3 JavaScript表单处理

（4） radio单选钮

其属性有：

**name:** 设定提交信息时的信息名称，对应文档中的**name**。

**Value：** 用以设定出现在窗口中对应档中**value**的信息

**length:** 对应该组单选钮选项的数目

**checked:** 指明该框的选择状态为**true**或**false**

对应的主要事件有：**onclick**

# 5.8文档对象模型DOM的基本概念和作用

## 5.8.1 DOM的基本概念

❖ DOM（Document Object Model）即“文档对象模型”，HTML DOM 定义了访问和操作 HTML 文档的标准方法, DOM 将网页内的元素与内容呈现为一个清晰、易读的树状模型，使得通过Javascript能够访问和修改HTML页面。

D（文档）：当创建了一个网页并将其加载到Web浏览器中时，DOM便根据这个网页创建出了一个文档对象。

O（对象）：一种独立的数据集合。例如文档对象，即是文档中元素与内容的数据集合。

M（模型）：文档对象的树状模型。在这个树状模型中，网页中的元素与内容表现为一个个相互连接的结点（node）。

# 5.8文档对象模型DOM的基本概念和作用

## 5.8.1 DOM的基本概念

DOM中的节点类型包括元素节点、文本节点及属性节点。

（1）元素节点：在HTML中，<body>、<DIV>、<a>等一系列标签，即是这个文档的元素节点。元素节点，组成了文档模型的语义逻辑结构。

（2）文本节点：包含在元素节点中的内容部分，如<p>标签中的文本等等。一般情况下，不为空的文本节点都是可见并呈现于浏览器中的。

（3）属性节点：元素节点的属性，如<a>标签的href属性与title属性等等，属性节点总是被包含于元素节点当中。

## 5.8.2 DOM的体系结构

在DOM中，HTML文档的层次结构被表示为一个树形结构。树的根结点是一个表示当前HTML文档的document对象，树的每个子结点表示HTML文档中的不同内容。

每个结点都由一个node对象表示，node对象提供了nodetype属性来表示结点的类型。

DOM为不同类型的结点提供了相应的接口，当知道一个结点为某种类型时，则可以使用相应的接口所定义的属性和方法。

# 5.8文档对象模型DOM的基本概念和作用

## 5.8.2 DOM的体系结构

节点间关系：

父子关系

兄弟关系

祖孙关系

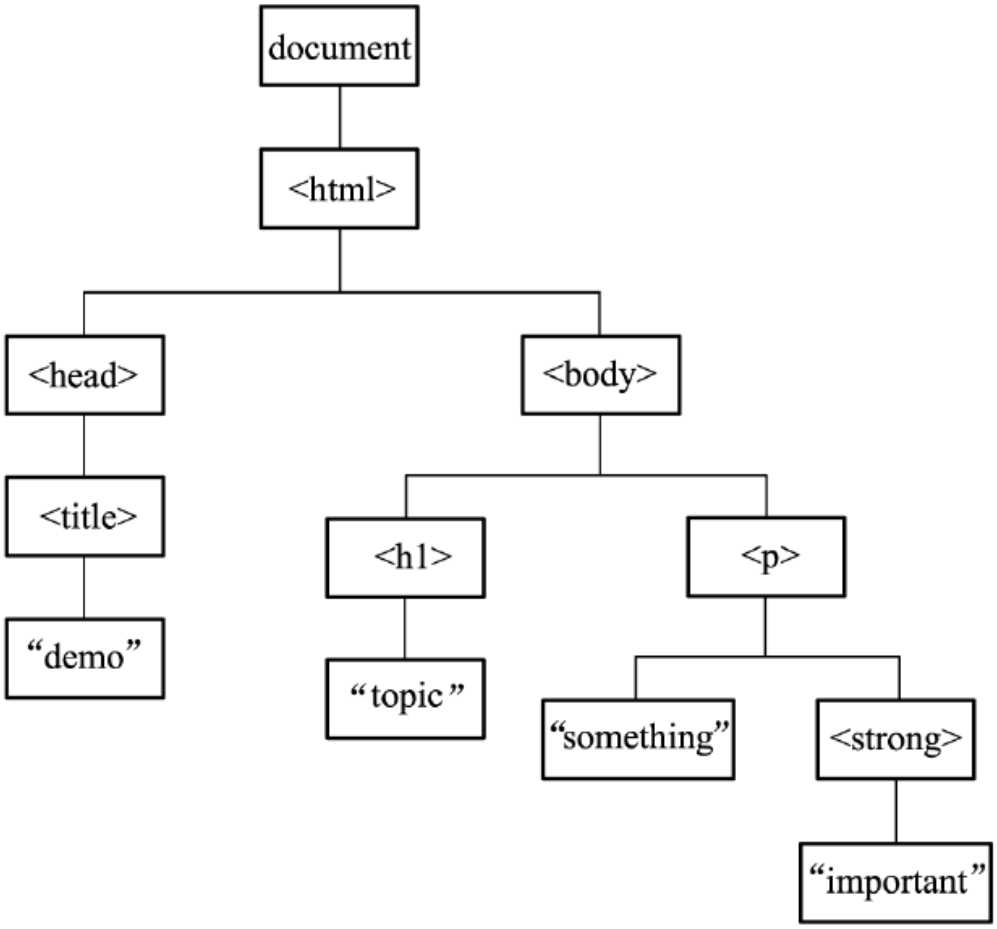



图5-5 DOM的体系结构

# 本章小结

客户机端的JavaScript脚本是通过标签<script>的内容嵌入到HTML文件中的。通过<script>标签的src属性设定脚本文件名，将脚本包含到HTML中。

BOM由多个对象组成，其中代表浏览器窗口的window对象是BOM的顶层对象，其他对象都是该对象的子对象。BOM提供了一组以window为核心的对象，实现了对浏览器窗口的访问控制。

在JavaScript中使用Form对象，可以实现动态的Web页面信息交互，但由于其源代码是公开的，使用JavaScript编程不适合于直接进行口令检测等安全性较高的内容。

DOM即“文档对象模型”，是基于语义的逻辑结构，DOM将网页内的元素与内容呈现为一个清晰、易读的树状模型，使得JavaScript能够访问和修改HTML页面。