第5章 客户端编程

【本章导读】

网页不仅仅包含HTML元素，通常还包含脚本程序，它们增加了页面的交互和计算能力，使网页更生动、功能更强大。脚本程序分为客户端脚本程序和服务端脚本程序两种，它们分别在Web浏览器中和Web服务器上执行。因此，在Web开发中，可分为前端开发和后端开发两个部分。前端开发是指客户浏览器端的开发，包括网页制作和客户端编程。后端开发则是指服务器端的开发，主要是业务逻辑处理和数据库编程。

本章介绍前端开发中的客户端编程问题，首先讲解Web浏览器和客户端脚本程序的关系，对浏览器的工作原理进行分析，它是理解脚本程序和脚本编程的关键。然后以JavaScript语言为例，讲解客户端脚本编程问题，包括JavaScript程序语言，对象及其操作，内部对象及函数，HTML文档对象模型以及库和jQuery。根据在Web开发中的具体需求，详细讲解了Web开发中有关表单编程中遇到的问题，包括数据的获取、可靠性验证、网页参数传递等，并提供了大量的实用代码，最后给出了几个综合案例。

【知识要点】

5.1节：计算机程序，程序设计语言，源程序，解释执行，程序编译，程序运行。

5.2节：浏览器脚本引擎，客户端脚本语言，JavaScript脚本语言，Jscript脚本语言，<script>标记，文件包含。

5.3节：保留字，标识符，小驼峰命名法，大驼峰命名法，序言性注释，描述性注释，数据类型，弱数据类型，变量，运算符，表达式，程序语句，顺序语句，分支语句，重复语句，函数，返回值。

5.4节：类，对象，new操作，点运算符，括号运算符，this指针。

5.5节：内置对象，字符串对象(String)，正则表达式，元字符，限定符，正则表达式对象(RegExp)，数学对象(Math)，日期对象(Date)，数组对象(Array)，预定义函数。

5.6节：浏览器对象模型BOM，window对象，location对象，history对象，screen对象，navigator对象。

5.7节：HTML文档对象模型DOM，document对象，body对象，HTML元素内存对象。

5.8节：AJAX技术，客户端和服务器的异步通信，页面局部刷新。

5.9节：JavaScript库，Prototype库，jQuery库，jQuery函数，jQuery插件。

5.10节：折叠式菜单，树状菜单，数据有效性验证。

5.1 计算机程序与程序设计语言

人们使用计算机,确切地讲,是使用计算机软件。计算机软件,即是计算机程序。在信息社会,程序不应该是一个深奥的专业术语,随着软件集成开发环境的发展,编程也不再是计算机专业人员的专利,程序应该是信息社会人们基本信息素养的一部分。

5.1.1 计算机程序设计语言

从专业的角度讲,计算机程序是一组可以在计算机CPU中执行的计算机指令序列。计算机程序设计语言是用于编写计算机程序的语言。计算机程序设计语言很多,但程序设计语言的基本成分都是相似的,主要包括:①基本符号,定义语言所使用的字符集、保留字、标识符、注释等;②数据和数据类型,对程序所处理的数据的描述;③常量和变量,声明程序中用到的存储数据的量;④表达式和运算符,数据运算规则;⑤基本语句,进行数据处理和流程控制,表达业务逻辑;⑥函数,实现模块化和结构化编程。

对于程序设计语言,可以从不同的方面进行分类。按照语言级别可以分为低级语言和高级语言。①低级语言与特定的计算机有关、运行效率高,但使用复杂,低级语言有机器语言和汇编语言。机器语言是基于机器基本指令集的,或者是操作码经过符号化的基本指令集。汇编语言是机器语言中地址部分符号化的结果。②高级语言,是一种更加符号化的语言,接近于自然语言,与计算机硬件没有紧密关系。

按编程思想分,计算机程序分为过程式程序设计语言和面向对象程序设计语言。过程式程序设计流行于20世纪90年代以前,自顶向下逐步求精的结构化程序设计是软件开发的主要方法,直到现在,这种结构化的程序设计思想仍然被广泛采用。Pascal,C,Basic,Fortran等高级语言很好地实现了结构化编程的思想,通过过程和函数(又称子程序),把一个复杂的问题划分成若干相对简单的子问题,如果子问题还比较复杂,再继续划分,最后将划分后的每个小问题编码成一个个的过程和函数,它们共同构成一个大的软件。

20世纪90年代,面向对象技术兴起,对人们近半个世纪来的软件开发思想产生了深刻变革。面向对象的思想将自然界中的物理对象映射为软件中的软件对象,建立了类和对象的概念。在面向对象技术中,不仅用对象实现了数据和操作的封装,还通过消息映射的方式在事件和函数之间建立关联,避免了传统过程式程序设计中函数显式调用的不足。键盘鼠标等事件的发生会发出消息,消息来激活函数,函数之间的联系不再是显式的调用,这样就降低了函数之间的耦合度。对于复杂系统,面向对象技术可以提高系统的可扩充性和代码重用的层次。当前流行的C++,Java都是典型的面向对象程序设计语言。

5.1.2 程序开发及其运行

用计算机程序设计语言编写的程序称为源程序,程序的执行分为解释执行和编译后执行两大类。解释执行就是在某个环境下逐条执行程序语句,当遇到程序错误时,则可能停止运行。目前,网页中的客户端脚本程序就是由Web浏览器中的脚本引擎解释执行的。编译型程序则是将源程序转换成可在CPU中运行的机器指令,通过操作系统调用来执行。

对于编译型程序,需要经过源程序编译、连接才能形成一个可在计算机操作系统上运行

的可执行文件。这个可执行文件被安装在计算机中，运行该程序，操作系统将把可执行程序文件调入计算机内存，在操作系统的调度下执行。编译型程序是计算机程序的主要形式，程序开发过程如图 5-1 所示。

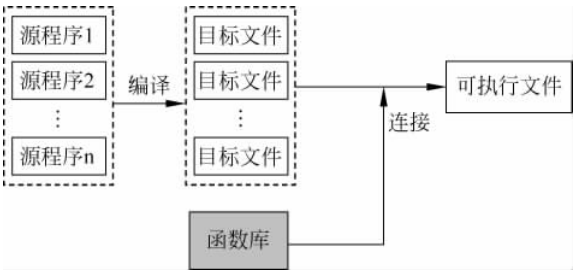

图 5-1　编译型计算机程序开发过程

编程人员利用开发工具(如 Visual C++，MyEclipse 等)来编写程序，即源程序。然后对源程序进行编译、连接操作，最终形成一个可执行文件。可执行文件(Executable File)指的是可以由操作系统进行加载执行的文件。在不同的操作系统环境下，可执行程序的呈现方式不一样。在 Windows 操作系统下，可执行程序可以是. exe 文件、. com 等类型文件。在 Linux 中的可执行文件格式为 ELF(Executable and Linkable Format)，在 Mac OS 中的可执行文件格式为 Mach-O 即 Mach Object 格式。

编译型程序一直是计算机软件开发的主流模式，我们生活中见到的大多数软件系统都是上述开发过程的结果。直到 Web 出现，一种基于 Web 服务的程序开发和运行模式，即浏览器/服务器(Browser/Server，B/S)模式开始出现并被广泛应用。B/S 模式表现出了更大的灵活性，从此基于 Web 的软件开发开始成为程序设计和应用的主流。

5.2　浏览器与客户端脚本程序

在互联网中，Web 浏览器是一种专门用于网页浏览的程序。用户在浏览器地址栏中输入网址，或在某个网页上单击一个超链接时，浏览器将和相应的 Web 服务器建立联系，发送网址给服务器，服务器将网页文件发送给用户浏览器，文件在浏览器中打开并显示。浏览器在对网页内容进行显示的同时，如果遇到客户端脚本程序，浏览器则执行脚本程序。

5.2.1　客户端脚本程序与脚本引擎

所谓客户端脚本程序，是指在客户浏览器中运行的程序。客户端脚本程序不需要事先编译，如果浏览器从服务器上下载的网页中包含客户端脚本程序，浏览器将对脚本程序代码进行解释执行。浏览器之所以能够解释执行网页中的客户端脚本程序，是因为浏览器中内置了脚本引擎模块，从而可以对 HTML 文档中的脚本程序进行分析、识别、解释并执行。

客户端脚本程序通常是用脚本程序语言书写的，脚本程序语言和传统的编译型程序设计语言(如 C/C++，Java 等)相比，在语法结构上类似，最大的不同是脚本程序不需要编译、连接过程，即不生成在操作系统下运行的可执行文件，而是直接在浏览器中，被浏览器解释执行。在解释执行过程中，如果程序存在错误，浏览器即停止程序的执行，并在浏览器窗口

的状态栏中显示“网页存在错误”的提示。

由于安全方面的原因,在浏览器设置中,可以使浏览器禁止脚本程序的运行。例如,在IE浏览器的“Internet选项”对话框中,包含“安全”选项卡,打开“自定义级别”,在安全设置列表中,可以在“活动脚本”中选择“禁用”“启用”或“提示”。如果选择“禁用”,则浏览器在打开网页时将不执行网页中的客户端脚本程序。

5.2.2 客户端脚本语言

在Web发展早期,用户提交一个表单,为了保证数据输入正确,浏览器和服务器之间可能需要进行多次交互。在网络速度很低的时代,这是不可接受的。人们自然想到的是,在数据提交以前,对用户输入进行有效性验证,即可解决该问题。

1995年末,网景公司(Netscape)[①]发布了导航者浏览器Navigator 2.0,首次设计实现了浏览器脚本程序语言,这样数据有效性验证的问题就可以在客户端解决了。由于当时和Sun公司开展合作,加之当时Java技术的流行,Netscape将其客户端脚本语言命名为JavaScript。它是一种直译式脚本语言,用于在HTML网页上使用,从而为HTML网页增加了编程功能,其解释器称为JavaScript引擎,为浏览器的一部分。

随后,为了取得技术优势,微软推出了JScript,CEnvi推出ScriptEase,与JavaScript一样可以在浏览器上运行。发展初期,JavaScript的标准并未确定,同期有Netscape的JavaScript,微软的JScript和CEnvi的ScriptEase三足鼎立。1997年,在ECMA(欧洲计算机制造商协会)的协调下,由Netscape、Sun、微软、Borland多家公司组成工作组,在JavaScript基础上制定了脚本语言标准,称为ECMAScript。

1. 脚本语言的组成

在ECMAScript标准中,规定了脚本语言的基本组成,包括三个部分:

(1) 语言语法和基本对象,这是一门程序设计语言的基本组成部分,包括:语法、类型、语句、关键字、保留字、运算符、内置对象等。

(2) 文档对象模型(DOM),描述处理网页内容的方法和接口。对于网页中的每一个标记,浏览器都为其在内存中创建一个对象,通过DOM编程来实现对网页的交互控制。

(3) 浏览器对象模型(BOM),描述与浏览器进行交互的方法和接口,实现在客户端脚本程序中对浏览器的访问和控制。

ECMAScript不与任何具体浏览器绑定,它只是描述了有关脚本程序语言所具有的通用属性,这些具体内容需要由具体的浏览器实现。

目前,在Web浏览器中,主要的客户端脚本语言有JavaScript和JScript,其中,JavaScript是最早的客户端脚本语言,是浏览器默认的脚本程序语言。不同的浏览器对脚本语言的实现不完全一样,这就导致同样的网页在不同的浏览器中打开的效果可能不同。

2. 脚本程序

根据HTML规范,在网页中书写脚本程序,脚本程序应该书写在< script >…</script >

① 网景(Netsape),美国著名浏览器厂商,1994年成立,同年12月,推出第一款商用浏览器导航者(Navigator),创始人之一就是马克·安德森(Marc Andreessen)。在微软进军浏览器市场以前,其市场份额一度达到90%,1998年11月24日,AOL收购网景,后来导航者(Navigator)浏览器逐步淡出市场。

标记对内。在网页中包含脚本程序的一般形式是：

```
<script type = " ">
    语句部分
</script>
```

Script 标记包括一个必选属性 type 和若干可选属性。必选属性 type 规定脚本的 MIME 类型(Multipurpose Internet Mail Extensions,多用途互联网邮件扩展),它是设定某种扩展名的文件用一种应用程序来打开的方式类型,当该扩展名文件被访问的时候,浏览器会自动使用指定应用程序来打开,多用于指定一些客户端自定义的文件名,以及一些媒体文件打开方式。对于 JavaScript 来讲,其 MIME 类型为“text/javascript”。

Script 标记还包括若干可选属性,例如：language 属性(用于设定脚本程序语言,也可以包含版本号。一般不设,由 type 属性设定),src 属性(外部脚本文件 URL),charset 属性(外部脚本文件中使用的字符编码),defer 属性(是否对脚本执行进行延迟,直到页面加载为止)等。

与传统的程序设计一样,在 JavaScript 编程中,也可以将一些公用的函数保存为独立的文件(扩展名为.js),然后在其他网页的头部(< head ></head >),把其他 JavaScript 文件包含进来,一般形式是：

```
<script src = "脚本文件 url"></script>
```

脚本程序可以出现在网页的头部,也可以出现在网页的文档体中,其中出现在文档头部的脚本程序通常是一些函数,这些函数只有在显式地调用时才被执行。在介绍具体的客户端脚本程序设计以前,先看两个包含 JavaScript 客户端脚本程序的简单网页示例。

【例 5-1】 编写一段脚本程序,检测所用 Web 浏览器对 JavaScript 脚本程序的支持情况。

说明：不同的浏览器对脚本程序的支持情况不同,下述代码可以检测浏览器对 JavaScript 脚本程序不同版本的支持情况。

用 Sublime Text 代码编辑器编辑 HTML 代码,代码清单如下：

代码清单(exa5-1.htm)：

```
<!DOCTYPE html>
<html>
<head>
<meta charset = "utf - 8">
</head>
<body>
<h1>JavaScript 检测</h1><hr>
<! -- JavaScript 支持性检测 -- >
<script type = "text/javascript">
      document.write("浏览器支持 JavaScript!<br><br>");
</script>
<! -- JavaScript 版本检测 -- >
<script type = "text/javascript1.0">
    document.write("浏览器支持 JavaScript 1.0<br>");
```

```
</script>
<script type="text/javascript1.1">
    document.write("浏览器支持 JavaScript 1.1<br>");
</script>
…
<script type="text/javascript1.5">
     document.write("浏览器支持 JavaScript 1.5<br>");
</script>

</body>
</html>
```

分别使用 IE10.0、Maxthon(遨游)、Mozilla Firefox 和 GoogleChrome 浏览器中打开上述网页文件 exa5-1.htm,在 IE 10.0 和 Maxthon(遨游)浏览器中,显示浏览器支持 JavaScript 1.1/1.2/1.3,因为 Maxthon 浏览器采用了 IE 内核,因此,两者对于 JavaScript 版本的支持一致。在 Mozilla FireFox 2.0 浏览器中,显示浏览器支持 JavaScript 1.0 到 JavaScript 1.5 的全部 Javascript 版本。而 GoogleChrome 浏览器只支持 JavaScript1.1/1.2/1.3。

与所有的软件开发一样,Web 页面中的程序也需要对程序进行结构化设计。在 JavaScript 中,同样提供了函数定义与函数调用功能,以支持结构化程序设计。由于浏览器浏览的 Web 页是顺序地从 Web 服务器调出,并由浏览器解释执行的,因此函数必须遵循先定义(一般放在<head>…</head>)后调用(在<body>…</body>内)的原则。

【例 5-2】 编写一个 JavaScript 函数,求一个正整数 n 的阶乘。

说明:模块化编程同样适用于脚本程序,函数可以写在 HTML 页面的任何位置,从良好的编程习惯上讲,函数定义通常写在<head></head>中。

代码清单(exa5-2.htm):

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script type="text/javascript">
function fact(n)
{
  if (n==0)
   return 1;
  else
  return n*fact(n-1);
}
</script>
</head>
<body>
<p>fact(5) =
<script type="text/javascript">
document.write(fact(5));
```

```
</script>
</body>
</html>
```

在浏览器中打开上述网页文件，在浏览器中显示：fact(5) = 120。

JavaScript 程序和一般的编译型程序不同，它没有一个操作系统调用的主程序（如 C 语言中的 main 函数），一个 JavaScript 函数定义时并不发生作用，只有在引用时（函数定义后的 document. write 语句）才被激活。

不同的浏览器和浏览器版本，支持的脚本语言版本也不相同。因此，在书写客户端脚本程序时，应该根据浏览器的种类和版本，使用合适的内置对象和浏览器对象，否则，如果程序中使用了一些高版本脚本语言包含的对象，则网页在不支持该版本的浏览器中，或低版本的浏览器中可能不能正常显示。

5.3 JavaScript 程序设计基础

在为数不多的客户端脚本语言中，JavaScript 脚本语言是使用最为广泛的客户端脚本程序设计语言，得到了所有浏览器的支持，是各种浏览器首选的默认脚本程序语言。与传统的 C/C++、Java 等程序设计语言不同，作为脚本语言，JavaScript 的特点是：①是一种弱类型的语言，对使用的数据类型未做出严格要求，语法类似 C/C++ 和 Java 语言，语言简单。②是一种基于对象的脚本语言，它不仅可以创建对象，还定义了一系列内置对象。③采用事件驱动方式，使客户端具有强大的编程能力。④跨平台性，因为 JavaScript 脚本程序在浏览器中执行，所以不依赖于操作系统，具有跨平台性和良好的兼容性。

5.3.1 JavaScript 基本符号

任何一种程序设计语言都有其自身的字符集和基本符号，它们按照程序设计语言的语法构成程序语句，然后语句再构成程序。

1. 基本字符

JavaScript 的基本字符有字母（a、b、…、z，A、B、…、Z）数字（0、1、…、9）和特殊符号（+、-、*、/、<、=、>等）三大类。

同 C/C++一样，JavaScript 中同样有些以反斜杠（\）开头来表示不可显示的特殊字符，通常称为控制字符，例如：'\n'表示换行（nextline，\13），'\r'表示回车（return，\10）。因为所有的 ASCII 码都可以用“\”加数字（一般是八进制数字）来表示。在 C 等计算机语言中，定义了一些字母前加“\”来表示那些不能显示的 ASCII 字符，如：\0，\t，\n，\r 等，因为后面的字符不再是它本来的 ASCII 字符的意思了，故称转义字符。

2. 关键字

关键字又称为保留字，它是由字母构成的具有固定含义的单词，如 var 代表变量说明，if 表示条件语句等。JavaScript 的关键字很多，可以参考 JavaScript 的专门书籍。

需要特别注意的是，JavaScript 是识别大小写的，在 JavaScript 中，关键字需要小写，如果书写有误，在打开网页时，在浏览器窗口状态栏将显示“JavaScript 脚本错误”的警告。

3. 标识符

表示常量、变量、类型和函数等名称的符号。标识符分为标准标识符和用户自定义标识符。标准标识符是表示标准常量、标准类型和标准函数等名称的符号。JavaScript 中的标准常量和标准函数见后续章节的介绍。

用户自定义标识符是指用于说明常量、变量、类型和函数等名称的符号。用户自定义标识符必须以字母开始,由任意的字母、数字和"_"组成。用户在命名标识符时应该有一定的命名规范:首先,用户自定义标识符不应该与标准标识符重名。第二,用户自定义标识符要尽可能反映它所代表对象的含义。如用 Name 代表名称比用 x 代表名称可读性更强。第三,如果是一个变量名,还应该尽可能反映变量的数据类型和作用域,比如用 nUserID 表示一个用户标识,最前面的小写字母 n 代表变量为整数类型。总之,好的命名规范可以提高程序的可读性,便于程序的维护。

目前常用的标识符命名法有小驼峰命名法和大驼峰命名法,所谓小驼峰命名法,就是第一个单词的首字母小写,其他单词首字母大写,例如:int myUserAccount。大驼峰命名就是所有单词首字母均大写,例如:public class UserAccount。

4. 注释

为了增加程序的可读性,一般在程序中增加注释语句,注释内容没有语法要求,其内容仅仅是一个提示作用。注释一般分为序言性注释和描述性注释两种类型。序言性注释出现在模块的首部,其内容一般包括有关模块功能的说明,界面描述,包括调用语句格式、所有参数的解释和该模块需调用的其他模块名等。还包括一些重要变量的使用、限制及其他信息。描述性注释嵌在程序之中,描述性注释又有功能性的和状态性的,功能性注释说明程序段的功能,通常可放在程序段之前,状态性注释说明数据的状态,通常可放在程序段之后。

在 JavaScript 中,注释以"//"字符引导,注释可以单独一行,也可以在语句行的后面。与 HTML 的注释(<!--….-->)不同,在服务器端处理脚本时,JavaScript 中注释将被删除,而不是被送到浏览器端。若需要客户端浏览器看到脚本中的注释,应该使用 HTML 注释将注释加进 HTML 页。此时,注释将返回给浏览器。

5.3.2 数据和数据类型

在程序设计中,所谓"数据类型",就是定义一段计算机内存空间的解析规则,说明变量就是要说明变量名称及其数据类型。数据类型决定了变量所占内存空间的大小,同时也决定了变量的取值范围。对于程序中声明的全局变量,程序在运行时,操作系统会为其分配空间。直到程序运行结束,所占用的内存空间被释放。

JavaScript 是一种解释性的程序设计语言,所写的脚本程序不需要编译和链接,它采用边读边执行的方式运行。在数据类型方面,JavaScript 提供了 4 种基本的数据类型用来处理数字和文本。JavaScript 提供的数据类型有:数值(整数和实数)、字符串型(用西文双引号""或西文单引号''括起来的字符),布尔型(true 和 false,均为小写)。

与传统的编译型程序设计语言不同,JavaScript 语言中采用弱类型的变量类型,没有预定义的类型名,变量不必事先声明其数据类型,可以在使用或赋值时确定其数据类型。此外,一个变量的类型在使用时还可以被改变。通常情况下,可以先给变量赋一个初值,通过

初值的类型来声明该变量的数据类型，这更加符合一般的程序设计思想，例如：

```
var x = "hello";
```

上述语句声明了一个变量 x，通过为 x 赋初值"hello"，将 x 作为一个字符串类型变量。在后面的语句中，可以给 x 赋一个整数，如：x＝100，这样变量 x 就成为整数类型了。在 JavaScript 中，上述变量类型的改变不会和其他程序设计语言一样出现赋值不相容的错误。

不同的数据类型之间可以进行转换，例如：将字符串转换为相应的整数，将整数转换为字符串数据等。这些转换被封装在相应的类或对象中，字符串到数值的转换函数可见 String 对象。如果是数值型转换为字符串，最简单的方法就是将一个字符串和数值做“＋”运算，结果即为字符串类型，例如：

```
var age = 18;
var agestr = age * 2 + "";
```

上述代码，第一个语句设置 age 为整数，第二个语句 agestr 则为字符串。通过整数和一个字符串，例如空串相加，即可得到一个字符串数据。

5.3.3 常量和变量

程序是对数据的处理，数据是通过常量和变量来表达的。一般情况下，系统不为常量分配内存空间。而变量在内存中则需要占据特定大小的存储空间，在程序运行过程中，通过变量赋值可以修改变量空间内存储的数值。

1. 常量和常量定义

常量是指在程序执行过程中，其值不发生变化的量。常量有字面常量和符号常量两种。字面常量就是一些数值或字符串，例如：3.14，"hello"等都是字面常量。根据数据类型的不同，常量可分为整数常量、实数常量、字符常量等。符号常量是指为一个常量起一个名字，即常量名，常量名是一个用户自定义标识符。例如：用 pi 代表圆周率 3.14。

常量命名有两方面的好处，首先恰当的常量名称可以增加程序的可读性；其次，使用常量名可以便于程序的维护。例如，可以命名常量 pi＝3.14，如果希望提高求解精度，可以修改 pi 的定义为 pi＝3.1416，这种修改只在一处进行，不会产生不一致的情况，而且修改简单。

在字面常量中，字符型常量使用得最为广泛，它是指使用单引号(')或双引号(")括起来的一个或多个字符。如'a'、"hello"、"5123"、"x＋y"等，其中单引号括起来的只能是单个字符，双引号括起来的为字符串，可以是单个字符，也可以是多个字符。

2. 变量和变量说明

所谓“变量”是指在程序执行过程中，其值发生变化的量。每一个变量都有一个变量名，对应一个特定的内存空间。变量有两个重要的属性，一个是数据类型，一个为操作运算。数据类型决定了变量所占内存空间的大小，也决定了数据的取值范围和操作运算。

在 JavaScript 中，变量命名的一般形式是：

```
var <变量名表>;
```

其中，var 是 JavaScript 的保留字，表明接下来是变量说明，变量名表是用户自定义标识

符,变量之间用逗号分开。与 C/C++不同,在 JavaScript 中,没有预定义的数据类型标识符,变量说明不需要说明数据类型。此外,变量也可以不说明而直接使用。

例如 var x,y; 则声明了两个变量,名称分别为 x 和 y,没有说明数据类型,也没有给变量赋值。

再如 var myName="John",则定义了一个变量 myName,同时赋予了它一个字符串值。在 JavaScript 中,变量可以不作声明,在使用时,数据的类型将确定变量的类型。

3. 变量作用域

变量作用域由声明变量的位置决定,决定哪些程序语句可访问该变量。在函数外部声明的变量称为全局变量,其值能被所在 HTML 文件中的任何脚本程序访问和修改。在函数内部声明的变量称为局部变量。局部变量只能被函数内部的语句访问,只对该函数是可见的,而在函数外部则是不可见的。当函数被执行时,局部变量被分配临时空间,函数结束后,变量所占据的内存空间被释放。

5.3.4 运算符和表达式

表达式是指将常量、变量、函数、运算符和括号连接而成的式子。根据运算结果的不同,表达式可分为算术表达式(结果为整数或实数)、字符表达式(结果为字符或字符串)和逻辑表达式(结果为逻辑值 true 或 false)。

JavaScript 提供了丰富的运算功能,包括算术运算、关系运算、逻辑运算和连接运算等。

1. 算术运算符

在 JavaScript 中,算术运算符有单目运算符和双目运算符。双目运算符包括:+(加)、-(减)、*(乘)、/(整除)、%(模除)、|(按位或)、&(按位与)、<<(左移)、>>(右移)等。单目运算符有:++(自加 1)、--(自减 1)、-(取反)、~(取补)等。

2. 关系运算符

关系运算又称比较运算,运算符有:<(小于)、<=(小于等于)、>(大于)、>=(大于等于)、=(等于)和!=(不等于)。

关系运算的运算结果为布尔值,如果条件成立,则结果为 true,否则为 false。

3. 逻辑运算符

逻辑运算符有:&(逻辑与)、|(逻辑或)、!(取反,逻辑非)、^(逻辑异或),运算结果为布尔值。

4. 字符串连接运算符

连接运算用于字符串操作,运算符为+(用于强制连接),将两个或多个字符串连接为一个字符串。

5. 三目操作符?:

三目操作符"?:"格式为:

```
操作数?表达式 1: 表达式 2
```

三目操作符"?:"构成的表达式,其逻辑功能为:若操作数的结果为 true(非 0 值),则表述式的结果为表达式 1,否则为表达式 2。例如 max=(a>b)? a:b;该语句的功能就是将 a,b 中的较大的数赋给 max。

5.3.5 语句

语句是对业务流程的描述，控制了程序执行的流程。所有的程序设计语言，无论是过程式程序设计语言还是面向对象的程序设计语言，其程序语句都可以分为三种类型，即顺序语句、分支语句和重复语句，使用这三种语句就可以描述用户的所有业务逻辑。

1. 顺序语句

从本质上讲，在一个程序中，语句的执行总是从上而下顺序执行的。在过程式程序设计语言中，这种顺序是程序本身所显式定义的。例如，C 语言中的函数调用，遇到函数调用，转去执行相应的函数，执行结束后返回。在面向对象的程序设计语言中，函数的调用变得更加复杂，它是在程序的运行过程中，由事件触发消息，由消息来激活函数。这样的事件驱动机制，使函数的调用变得不再和过程式程序设计中的显式调用那么清晰，但这种消息影射(Message Map)机制降低了函数之间的耦合度，大大增强了软件系统的可维护性。在函数内部，语句仍然是从上向下顺序执行的。

无论是 C、C++、Java，还是 JavaScript，由于语句语法的需要，有时候需要将多个语句看作逻辑上的一个语句，此时需要将这多个语句用一对花括号“{”和“}”括起来，语句之间用分号分开，称为语句块。在语句块内部，语句从上而下顺序执行。

2. 分支结构

在业务描述中，有些业务的执行是有条件的。要表达这样的业务流程，在程序设计语言中需要通过分支语句来实现。在 JavaScript 中，实现分支结构的语句有三种，它们是条件判断语句 if 语句、if…else…语句和开关语句 switch 语句。

(1) if 语句

if 语句的一般形式是：

```
if (<条件表达式> )
    <语句>;
```

if 语句首先计算条件表达式的值，若计算结果为 true，或非 0 值的数，包括正数或负数，则执行语句部分，否则执行 if 语句下面的语句。

if 语句逻辑功能如图 5-2 所示。

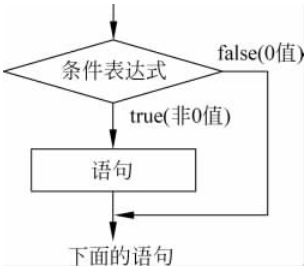

图 5-2 if 语句逻辑功能图

语句部分逻辑上是一个语句，如果语句部分需要多个语句来实现，应将这多个语句用“{”和“}”括起来，形成语句块，作为逻辑上的一个语句。

例如：

```
if  (a>=b)
{
    max=a;
    min=b;
}
```

(2) if…else 语句

在需要多种条件判断的业务逻辑中，用 if 语句来描述是麻烦和低效的。对于有两种分

支的情况,一般可以使用 if…else 语句来描述。if…else 语句的一般形式是:

```
if (<条件表达式> )
  <语句 1 >;
else
  <语句 2 >;
```

if…else 语句首先计算条件表达式的值,若为 true,或非 0 值的数,则执行语句 1,否则,执行语句 2,逻辑功能如图 5-3 所示。

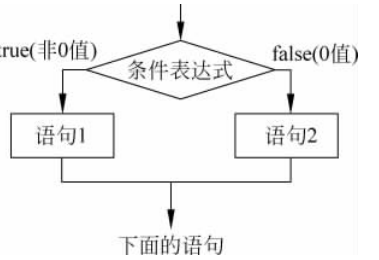

图 5-3 if…else 语句逻辑功能图

在存在两种分支的情况下,if…else 语句可以很好地描述这种逻辑。在多种分支的情况下,可以使用 if…else 语句的嵌套进行描述,也就是说,语句 1 或语句 2 本身也是 if…else 语句。语句嵌套虽然可以描述多种分支的业务逻辑,但嵌套语句降低了程序的可读性。为此,对于多种分支的情况,JavaScript 提供了 switch 语句。

(3) switch 语句

对于有三种以上分支的情况,采用 if…else 语句嵌套降低了程序的可读性。switch 语句提供了 if…else 多层嵌套结构的一种变通形式,可以从多个语句块中选择执行其中的一个。switch 语句提供的功能与 if…else 语句类似,但是可以使代码更加简练易读。

switch 语句一般形式是:

```
switch (<条件表达式>)
{
    case 表达式 1
         语句块 1;
         [break;]
    case 表达式 2
         语句块 2;
         [break;]
          ⋮
    case 表达式 n
         语句块 n;
         [break;]
    [default:
       语句块 n + 1;
    ]
}
```

在 switch 语句的开始是一个条件表达式,该条件表达式可以是一个整数、小数或字符串表达式。条件表达式的计算结果将与结构中每个 case 分支表达式值比较。如果匹配,则执行与该 case 关联的语句块。在语句块中,如果包含 break 语句,此时,则退出 switch 语句,否则,将继续下面的 case 匹配。switch 语句的逻辑功能如图 5-4 所示。其中虚线或框为可选语句及流程。

3. 重复结构

在业务流程中,有些部分是需要反复执行的。这需要通过程序设计语言中的重复语句

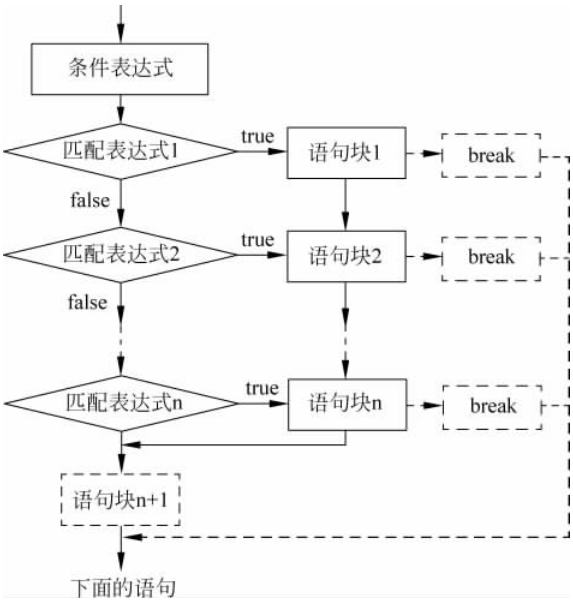

图 5-4　switch 语句逻辑功能图

来描述。重复语句总是由循环体和循环终止条件两部分组成，在 JavaScript 中有三种形式的重复语句。

（1）while 语句

while 语句是先判断循环终止条件，然后再执行循环体，因此循环体可能一次也不被执行。while 语句的一般形式是：

```
while (<条件表达式>)
    <语句>;
```

首先计算条件表达式的值，若计算结果为 true，或非 0 的数，则执行循环体语句，然后无条件地返回 while 语句开始，继续计算条件表达式的值。如果条件表达式计算的结果为 false，则结束循环，执行 while 循环语句下面的语句。逻辑功能如图 5-5 所示。

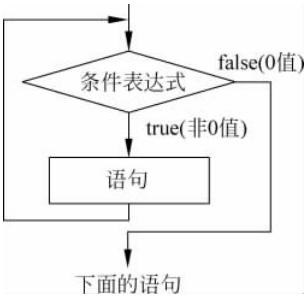

图 5-5　while 语句逻辑功能图

（2）do…while 语句

与 while 语句相似，不同的是 do…while 先执行循环体，然后再判断循环终止条件。因此循环体至少执行一次。do…while 语句的一般形式是：

```
do{
    <语句>
}while (<条件表达式>);
```

首先执行循环体，然后计算条件表达式的值，若计算结果为 true，或非 0 的数，然后无条件地返回 do…while 语句开始，继续执行循环体。直到条件表达式计算的结果为 false，则结束循环。逻辑功能如图 5-6 所示。

(3) for 语句

一般形式为：

```
for (表达式 1; 表达式 2; 表达式 3)
    <语句>;
```

执行过程如下：

Step1：计算表达式 1。

Step2：计算表达式 2，如果结果为 true，或非 0 值，则执行循环体，然后转 Step3。否则，结束循环。

Step3：计算表达式 3。

Step4：无条件转 Step2。

Step5：循环结束，执行 for 语句下面的语句。

逻辑功能如图 5-7 所示。

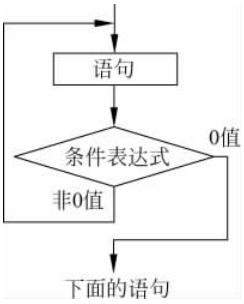

图 5-6　do…while 语句逻辑功能图

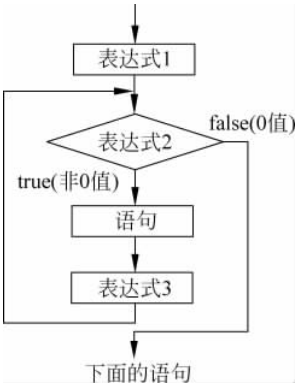

图 5-7　for 语句逻辑功能图

在 for 语句中，表达式 1、表达式 2 和表达式 3 中可以省略其中的一个或多个，但它们之间的分号不能省略。例如，可以写下面的 for 语句形式：

```
for (; ;)
{
…
}
```

上述 for 语句是一个死循环，在循环体中必须包含 break 语句，以结束循环的执行。另外，循环体的内部也可以包含循环语句，构成多重循环，即循环的嵌套。需要特别注意关键字要小写，例如 for 不应该写成 For。

4. break 和 continue 语句

与 C/C++ 语言相同，使用 break 语句使得程序执行从 for 语句或 while 语句中跳出，continue 使得跳过循环内剩余的语句而进入下一次循环。

5.3.6　函数

在结构化程序设计中，函数是实现结构化程序设计的主要手段，它把一个系统中需要反

复多次执行的部分定义为一个函数。如判断一个数是否为素数、求两个数的最大公约数等。在 JavaScript 中，已经给出了许多标准函数。同时也允许用户自己定义函数。

在 JavaScript 中，函数是以 function 开头定义的，由函数头部和函数体构成，一般形式为：

```
function <函数名> (<形式参数表>)
{
    变量说明部分;
    语句部分;
    return(<表达式>);
}
```

在 JavaScript 函数定义中，函数名后有形式参数表，这些形式参数变量可能是一个或几个，在 JavaScript 中可通过函数名.arguments.Length 来检查参数的个数。在函数体中，可以分为变量说明部分和语句部分，变量说明用于说明该函数所要处理的数据，语句部分是对变量的处理。在 JavaScript 中，由于数据类型采用弱数据类型，变量说明部分可以省略不写。

在函数体的语句部分通常包含一个返回函数值的语句，即 return 语句，返回函数值。return 语句不一定是语句部分的最后一条语句，但逻辑上该语句会被执行。如果函数体不包含 return 语句，则默认的函数返回值为 true。

5.4 类与对象

在软件系统开发中，面向对象程序设计(Object Oriented Programming，OOP)是目前最主流的程序设计思想，大部分的软件开发工具也是面向对象的，例如 C++、Java 等。面向对象的思想将问题域中的每一个实体都映射为软件系统中的一个对象，通过类和对象，实现了数据和数据操作的封装，极大地增强了系统的可维护性和可扩展性。

5.4.1 类与对象的概念

类(Class)和对象(Object)是面向对象程序设计的灵魂。所谓“类”，它是指一种包含成员变量和成员函数的数据结构。类和传统的过程式程序设计中的数据类型相似，它不占用内存空间，主要目的是用于声明对象。对象是用类来声明的数据结构，如果将类比作数据类型，对象就是相应数据类型的变量，在内存中分配特定大小的空间，存储数据，类决定了对该存储空间的语义解析。

JavaScript 并不是一个完整的面向对象的程序设计语言，它不提供关于对象的抽象、封装、继承、派生、多态等功能。它是一种基于对象的程序设计语言，预定义了一组内置对象和浏览器对象。同时，对 HTML 文档，浏览器也创建 DOM 对象。通过这些对象操作，从而分享面向对象程序设计带来的好处。

因为 JavaScript 是一种基于对象的程序设计语言，理论上讲，JavaScript 并没有定义类的功能。在 JavaScript 中，所谓定义“类”，其实是一种模拟的自定义类型。有两种常见的形式，介绍如下：

(1) 利用函数构造类

该定义类的方式和“函数”定义的语法相同,而且这样的函数称为该类的构造函数。用户用函数定义来定义类,然后用 new 语句创建该类的一个实例(对象)。一般形式是:

```
function className(< prop1,prop2,...>)
{
    this.prop1 = prop1;              //属性
    this.prop2 = prop2;
    ...
    this.method1 = FunctionName1;  //方法,需要定义对应的函数
    this.method2 = FunctionName2;
    ...
}
```

上述定义即定义了一个名为 className 的类,它包含的属性是 prop1,prop2,...,包含的成员函数是 method1,method2,...。

(2) 利用原型 prototype 定义类

在 JavaScript 中,类是以函数的形式来定义的。每个类对象都包含一个 prototype 属性成员对象。用户可以向对象的 prototype 属性添加属性和方法。

例如:为 JavaScript 内置对象 Array 添加方法:

```
< script lanuage = "javascript">
Array.prototype.max = function()
{
    var i, max = this[0];
    for (i = 1; i < this.length; i++)
    {
      if (max < this[i])
        max = this[i];
    }
    return max;
};
</script >
```

删除 String 对象两侧的空格,代码如下:

```
String.prototype.trim()
{
    return this.replace(/(^\s+)|(\s+ $ )/g, "");
}
```

除了可以为内置对象添加属性和方法外,还可以为用户定义的类添加方法,示例代码如下:

```
function TestObject(name)
{
    this.m_name = name;
}
TestObject.prototype.ShowName = function()
{
```

```
    alert(this.m_name);
};
```

上述代码定义了一个类 TestObject，包含一个属性 m_name，一个方法 ShowName()。

【例 5-3】 定义一个 JavaScript 类，包含一个成员变量，三个成员函数，并用不同的方法定义成员函数，比较它们的不同。

分析：在 JavaScript 中，可以使用 function 定义类。对类成员函数可以用不同的方法定义，不同方法定义的函数不同。

代码清单：

```
<html>
<head>
<meta charset="utf-8">
<script language="javascript">
function f1()
{
  return this.m_name;
}
////////////////////////////////////////////////////////////////////////////
//定义 TestObject 类
function TestObject(name)
{
  this.m_name = name;
  //成员函数定义方法 1
  this.f1 = f1;
  //成员函数定义方法 2
  this.f2 = function()
  {
    document.write("this.f2()<br>");
  }
}
//成员函数定义方法 3
TestObject.prototype.ShowName = function()
{
  document.write(this.m_name);
}
</script>
<head>
<body>
<script language="javascript">
a = new TestObject("Obj A");
b = new TestObject("Obj B");
a.ShowName();
b.ShowName();
document.write(a.f1 == b.f1);
document.write(a.f2 == b.f2);
document.write("<br>");
```

```
document.write(a.ShowName == b.ShowName);
</script>
</body>
</html>
```


在上述代码中,在页面的头部定义了一个 JavaScript 类 TestObject,用三种方法定义了三个成员函数 f1()、f2()和 ShowName()。在页面的< body >体内,声明了两个对象 a 和 b。在浏览器中打开该页面,显示结果如下:

```
Obj A
Obj B
true false true
```

从上述输出结果可以看出,三种方法定义的成员函数不同,如果在类的定义中嵌入函数定义(方法 2),不同的实例对象将拥有独立的成员函数副本。使用 prototype 定义的成员函数(方法 3),所有对象拥有同一个成员函数。

此外,不同方式定义的成员函数,JavaScript 解释引擎搜索的顺序也不相同,具体过程是:JavaScript 解释引擎在处理使用"."或"[]"操作符引用的对象的属性和方法时,先在对象本身的实例(this)中查找,如果找到就返回或执行。如果没有查找到,就查找对象的 prototype 成员对象,查看是否定义了被查找的对象和方法,如果找到就返回或执行,如果没有查找到,对于属性就返回 undefined,对于方法就返回 runtime error()。

5.4.2 对象操作

无论是 C++、Java 等面向对象的程序设计语言还是 JavaScript 这样基于对象的程序设计语言,定义类的目的是为了说明对象,对象实现数据的存储,类只是定义了对象的数据结构和内存解析规则。

1. 创建对象

创建对象就是按照类的定义在内存中分配一段空间,并为该空间命名。在 JavaScript 中,使用 new 运算符可以创建一个新的对象。创建对象的一般形式是:

```
myobj = new calssName(参数表);
```

其中,myobj 为新建对象的名称,calssName 是类的名称,参数表用于激活类的相应构造函数,从而为类中的成员变量赋值。

例如,创建一个日期对象,代码为:

```
myDate = new Date();
myBirthday = new Date("May 11,1975 6:15:00");
```

这样就创建了两个新的日期对象 myDate 和 myBirthday,myDate 对象取当前的计算机时钟作为其日期值,而第二个 Date 对象 myBirthday 被赋值一个具体的日期和时间。

2. 访问对象属性和方法

访问一个对象,就是访问对象的属性和方法。对象属性和方法的引用可以使用点(.)操作符或括号([])操作符,一般形式为:

```
对象名.属性名|方法名
```

或：对象名[属性名|方法名]

其中，点(.)操作符是最常使用的形式，类似于结构体。而括号([])操作符常用于数组中元素的访问。

3. for…in 语句

格式如下：for(对象属性名 in 对象名)

顺序输出对象各个属性的值，如果是方法，则输出对应的函数代码。例如：

```
function showData(object)
{
    for(var prop in object)
        document.write(object[prop]);
}
```

使用该函数时，在循环体中，可以不知道对象属性的个数，for 可以自动地将属性取出来，直到最后为止。

4. this 关键字

关键字 this 是对当前对象的引用，在 JavaScript 中，对象的引用是多层次的，往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，为此 JavaScript 提供了一个用于指定当前对象的指针 this。

5.5 JavaScript 内置对象及全局函数

每一种程序设计语言，都可以分成语言语法和标准库两个部分。语法部分通常用于定义语言的基本语法规范，包括字符集、保留字、数据和数据类型、程序语句等内容。标准库则是指为用户提供的一组常用函数库或类库。对于结构化程序设计语言，例如 C 语言，标准库通常是一组标准的函数库，包含了大量的标准函数。如果是面向对象的程序设计语言，例如 C++、Java 等，标准库则是一组标准类库，包含了大量的预定义类或模板类。使用标准函数库或类库，可以提高用户的编程效率和保证程序代码质量。

由于 JavaScript 是一种基于对象的程序设计语言，它具有面向对象程序设计语言的特点，但又不完全是面向对象的程序设计。在类库设计方面，JavaScript 不仅提供了一组标准类库，同时还提供了一组常用的对象和全局函数。而在面向对象的程序设计思想中，尽量不使用全局对象变量，所有的函数都应该隶属于一个类。不同的 JavaScript 版本，包含的内部对象和函数也不一样，本节介绍 JavaScript 中常用的内部对象和函数。

5.5.1 字符串对象 String

在 JavaScript 中，每个字符串都是一个 String 对象。使用 String 对象时，不需要像一般自定义对象一样用 new 关键字在内存中创建对象，而是可以直接将一个字符串赋给一个变量。字符串对象封装了 JavaScript 中的字符串以及相关的操作。

字符串对象的生成十分简单，而且是隐式的，不使用 new 关键字，例如：

```
var myStr = "Hello";
```

这样，myStr 就是一个 String 对象了，一个变量被声明为字符串对象之后，它就拥有了

这个对象类的属性和方法,可以和一般对象一样,使用对象的方法,取得对象的属性。

1. String 对象的属性

字符串对象的属性只有一个,这就是 length(长度)属性,返回字符串的长度。在 UTF-8 编码中,每一个汉字和英文字符一样,都属于一个字符。

例如:

```
<script type = "text/javascript">
    var myStr = "Hello 你好";
    document.write(myStr.length);
</script>
```

上述代码在浏览器中输出字符串"Hello 你好"的长度为 7。

2. String 对象常用方法

字符串对象是使用最为频繁的对象,因此包含更多的成员函数,常见的 String 对象成员函数见表 5-1。

表 5-1 String 对象常用成员函数

函数及一般形式	功 能
charAt(pos)	取字符函数,返回字符串对象中下标为 pos 的字符(字符串下标 pos 为大于等于 0,且小于字符串长度的整数)
substring(pos1,pos2) substring(pos)	第一种格式中,返回下标从 pos1 到 pos2 之前的字符串(不含下标是 pos2 的字符)。第二种格式返回下标为 pos 开始直到字符串结束的所有字符
indexOf(subStr) indexOf(subStr,StartPos)	subStr 是一个待查找的字符或者字符串,可以是常量,也可以是变量。在省略 StartPos 的情况下,此函数将从字符串的第一个字符开始查找;当 StartPos 参数存在的情况下,这个函数从字符串中下标为 StartPos 的字符开始查找。当 StartPos 超过字符串的长度时,返回−1。当所希望查找的字符串找不到时,返回−1。当字符串中有两个以上的待查找字符串,则返回被搜寻字符串中位置在最前面的待查字符串的下标位置
lastIndexOf (subStr) lastIndexOf (subStr,StartPos)	同 indexOf()函数类似,只是查找是从字符串尾部往前查找
match(searchvalue) match(regexp)	在字符串内检索指定的值,或找到一个或多个正则表达式的匹配。参数 searchvalue 规定要检索的字符串值。参数 regexp 为正则表达式。如果没有找到任何匹配的文本, match()返回 null。否则,返回一个数组,其中存放了与它找到的匹配文本有关的信息
replace(str1,str2)	将字符串中的字串 str1 用 str2 替换,同时 replace()支持正则表达式,它可以按照正则表达式的规则匹配字符或字符串
toLowerCase()	将字符串中所有的字母变为小写字母,返回变化后的结果。但是,原字符串内的大小写不变
toUpperCase()	将字符串中所有的字母变为大写字母,返回变化后的结果。但是,原字符串内的大小写不变
split(separator, limit)	将字符串分割为字符串数组,并返回此数组。Separator 为分隔符,limit 为可选参数,表示分割的次数,如果无此参数为不限制次数
length()	返回字符串的长度,所谓字符串的长度是指其包含的字符的个数

在 JavaScript 中，提供了大量的字符串处理函数，功能非常强大，下面是几个简单的例子。例如，在 HTML 的多行文本框输入中，包含了回车换行符，如果要保存到数据库中，则回车换行无法保存。如果要将数据库内容显示，则没有换行，因此，在保存数据库时可以将文本框中的回车换行替换为 HTML 的< br >标记，代码如下：

```
//设字符串保存在变量 str 中，删除其中的回车(\r)、换行符(\n)
str = str.replace(/\r/g,"< br >");
str = str.replace(/\n/g,"");
```

上述代码使用了正则表达式，“\r”表示回车字符(\13)，“/g”表示全部，即将 str 中全部的“\r”替换为< br >。第二行则表明将全部的换行“\n”(\10)，替换为空，即删除。如果要将全部的字符“a”替换为“A”，可写为 str＝str. replace(/a/g,"A")；

字符串的打散也是常用的操作之一，例如：

```
var str1 = "aa;bb;cc";
var str2 = "hello";
s1 = str1.split(";");
s2 = str2.split("");
```

则得到一个字符数组 s1＝["aa","bb","cc"]，s2＝["h","e","l","l","o"]。

与 split 对应的函数是 Array 对象的 join(delimiter)方法，使用给定的分隔符将一个数组合并为一个字符串，例如：

```
var aa = new Array("jpg","bmp","gif","ico","png");
var str = aa.join("|");
```

则字符串 portableList 为“jpg|bmp|gif|ico|png”。

5.5.2 正则表达式对象 RegExp

在字符串处理中，经常会用到查找某种特定模式的字符串，或者是验证某个字符串是否符合特定的模式。例如，检查一个字符串是否为有效的邮箱地址，检查用户账户是否是字母和数字的组合等。用来描述这种复杂规则的表达式称为正则表达式。在 JavaScript 中，RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具。使用正则表达式，可以大大减少字符串处理的编程工作量，使数据有效性验证更加准确和高效。

1. 正则表达式的概念

正则表达式(Regular Expression)的概念最初出现于理论计算机科学的自动控制理论和形式语言理论中，用于对模型和规则的一种形式描述。1950 年，正则表达式的概念被应用于 UNIX 的编辑器工具软件，随后被普及开来。许多程序设计语言都支持利用正则表达式进行字符串操作。

正则表达式是一种由普通字符和特殊字符(元字符)组成的字符串匹配模式，在正则表达式中，元字符(Metacharacter)是拥有特殊含义的字符。正则表达式只能使用"/"开头和结束，不能使用双引号，因为双引号是字符串对象的表示方法。

在 JavaScript 中，正则表达式中可以使用的元字符及其含义见表 5-2。

表 5-2 JavaScript 正则表达式元字符

元字符	含　义	元字符	含　义
.	匹配除换行符(\n)以外的任意字符	\0	匹配 NULL
\d	匹配数字(0..9)	\r	匹配回车符
\D	匹配非数字字符	\n	匹配换行符
\s	匹配空白字符	\t	匹配制表符
\S	匹配非空白字符	\v	匹配垂直制表符
\w	匹配一个字母、数字或下划线画单词字符	\f	匹配换页符
\W	匹配非单词字符	[ ]	匹配方括号之间的任何字符,字符范围可以用"-"连接,例如[abcd]可以写为[a-d]
\b	查找处在单词的开始或结束的匹配	[^]	匹配不在方括号之间的任何字符
\B	查找不在单词开始或结束处的匹配	(和)	指定子表达式,用于分组

在上述表格中,所谓"匹配",就是指字符串符合某种条件(正则表达式),通常是指这个字符串里有一部分(或几部分)能满足正则表达式给出的条件。另外,空白字符是指空格符(space character),回车符(carriage return character),换行符(new line character),制表符(tab character),垂直制表符(vertical tab character),换页符(form feed character)。

如果要查找元字符本身的话,比如查找字符"."或"*",应使用转义字符。转义字符使用"\"开始,用于取消紧跟在后面的字符的特殊意义。因此,要查找字符"."或"*",应该使用\.和*。要查找"\"字符本身,可使用\\。

例如:myfile\.dat 匹配 myfile.dat,d:\\GPMS3 匹配 d:\GPMS3。

在正则表达式中,还会遇到重复的情况,这需要使用量词(限定符)来指定重复的数量,常用限定符见表 5-3。

表 5-3 常用限定符

量词/语法	说　明
n*	匹配任何包含零个或多个 n 的字符串。例如:/lo*/g,在字符串中查找包含 0 个或多个"o"的字符串
n+	匹配包含至少一个 n 的任何字符串
n?	量词,匹配任何包含零个或一个 n 的字符串
n{x}	匹配包含 x 个 n 的序列的字符串
n{x,y}	匹配包含 x 或 y 个 n 的序列的字符串
n{x,}	匹配包含至少 x 个 n 的序列的字符串
^n	匹配任何开头为 n 的字符串。例如:/^You/g,查找所有以 Y 开头的字符串 You
n$	匹配任何结尾为 n 的字符串
?=n	匹配任何其后紧接指定字符串 n 的字符串。例如:/is(?=ok)/g,则查找所有的后面是"ok"的字符串"is"
?!n	匹配任何其后没有紧接指定字符串 n 的字符串

2. 创建 RegExp 对象

对于正则表达式,可以在有关的字符串对象中直接使用,也可以创建 RegExp 对象,然后在字符串操作中使用。前者使用直接量语法,一般形式为:

```
/pattern/attributes
```

如果要创建 RegExp 对象，一般形式为：

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

其中，参数 pattern 是一个字符串，指定正则表达式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，gi 表示全局并区分大小写的匹配。

例如：str＝str.replace(/\r/g,"< br >")；将字符串 str 中的所有回车字符(\r)全部用< br >替换。

对于下述代码，则创建一个 RegExp 对象。

```
<script type = "text/javascript">
function checkemail(str)
{
     //在 JavaScript 中，正则表达式只能使用"/"开头和结束，不能使用双引号
    var Expression = /\w + ([ -+ . ']\w + ) * @\w + ([ - . ]\w + ) * \. \w + ([ - . ]\w + ) * /;
    var objExp = new RegExp(Expression);
    if (objExp.test(str) == true)
       return true;
    else
       return false;
}
</script >
```

正则表达式在一开始看上去是比较麻烦的，其实，一个正则表达式是由元字符和限定符构成的串，元字符的后面跟限定符，只要理解这两类符号的含义，正则表达式的含义也就不难理解了。例如，\w＋表示至少一个字母、数字或下画线字符。

3. RegExp 对象属性和方法

RegExp 对象是 JavaScript 中提供的对字符串进行匹配、替换等操作的对象，和其他对象一样，也是由一系列的属性和方法构成的，见表 5-4。

表 5-4　RegExp 对象属性

属　　性	说　　明
global	查看给定的正则表达式是否执行全局匹配。如果 g 标志被设置，则该属性为 true，否则为 false
ignoreCase	如果设置了"i"标志，则返回 true，否则返回 false
multiline	查看正则表达式是否以多行模式执行模式匹配。在这种模式中，如果要检索的字符串中含有换行符，^和 $ 除了匹配字符串的开头和结尾外还匹配每行的开头和结尾
lastIndex	保存上一次匹配文本之后的第一个字符的位置。上次匹配的结果是由方法 RegExp.exec()和 RegExp.test()找到的，它们都以 lastIndex 属性所指的位置作为下次检索的起始点。找不到可以匹配的文本时，lastIndex 属性重置为 0
source	返回模式匹配所用的文本。该文本不包括正则表达式直接使用的定界符，也不包括标志 g、i、m

RegExp 对象方法见表 5-5。

表 5-5 RegExp 对象方法

方 法	说 明
test(str)	检测字符串 str 是否匹配某个模式。如果 str 中含有与 RegExpObject 匹配的文本,则返回 true,否则返回 false
exec(str)	检索字符串中正则表达式的匹配。参数 str 为要检测的字符串。返回一个数组,存放匹配结果。如果未找到匹配,则返回 null
compile (exp,mod)	在脚本执行过程中编译正则表达式。参数 exp 为正则表达式,mod 规定匹配的类型。"g"用于全局匹配,"i"用于区分大小写,"gi"用于全局区分大小写的匹配

例如,有一段示例代码如下:

```
<script type = "text/javascript">
    var str = "Hello World!";
    var patt = /lo * /g;
    document.write(str.match(patt));
</script>
```

在上述代码中,使用了 String 对象的 match 方法,查找字符串中匹配的子字符串。输出结果为:l,lo,l

例如:(\d{1,3}\.){3}\d{1,3}是一个简单的 IP 地址匹配表达式。其中,\d{1,3}匹配一到三位的数字,(\d{1,3}\.){3}匹配三位数字加上一个英文句号,这个整体被定义为一个分组,然后分组重复 3 次,最后再加上一个一到三位的数字(\d{1,3})。这是一个简单的 IP 地址模式匹配,当然,IP 地址的每一个数字都是小于 255 的,可以进一步写出更加准确的验证 IP 地址的正则表达式。

在书写正则表达式时,还经常用[]来指定一个范围,例如:[0-9]代表的含义与\d 完全一致,表示一位数字。[a-z0-9A-Z_]等同于\w。例如,对于下述代码:

```
<script type = "text/javascript">
    var str = "Hello_你好!";
    var patt = /\w/g;
    document.write(str.match(patt));
</script>
```

输出结果为:H,e,l,l,o,_

可见元字符"\w"只是匹配一个字母、数字或下画线"_",不能匹配汉字等其他字符。

例如:正则表达式 0\d\d-\d\d\d\d\d\d\d\d,它匹配这样的字符串:以 0 开头,然后是两个数字,然后是一个连字号"-",最后是 8 个数字(即中国的区号为 3 位的电话号码)。其中"\d"是元字符,匹配一位数字(0,1,2,…)。"-"不是元字符,只匹配它本身——连字符或者减号。为了避免重复,也可以这样写这个正则表达式:0\d{2}-\d{8}。"\d"后面的{2}、{8}表示前面的"\d"必须连续重复匹配的次数为 2 次(8 次)。

4. 常用正则表达式

正则表达式通常用于表单中的字符串处理、表单数据的有效性验证等,实用高效。下面是一些常用的正则表达式。

(1) 匹配中文字符的正则表达式：[\u4e00-\u9fa5]。

(2) 匹配空白行的正则表达式：/\n\s*\r/，元字符\s 匹配一个空白字符，后跟限定词*表示任意个数的空白字符。

(3) 匹配 HTML 标记的正则表达式：/<(\S*?)[^>]*>.*?</\1>|<.*?/>/。

(4) 匹配首尾空白字符的正则表达式：/^\s*|\s*$/，非常有用的表达式，可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等)。

(5) 匹配 E-mail 地址的正则表达式：/\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/，常用于表单验证。

(6) 匹配网址 URL 的正则表达式：/[a-zA-z]+://[^\s]*/。

(7) 匹配账号是否合法，以字母开头，允许 5～16 字节，允许字母、数字、下画线，则符合上述要求的正则表达式为/^[a-zA-Z][a-zA-Z0-9_]{4,15}$/。

(8) 匹配国内电话号码：/\d{3}-\d{8}|\d{4}-\d{7}|\d{4}-\d{8}/，匹配形式如 010-11112222、0531-8836111 或 0531-88361111。

(9) 匹配中国邮政编码：/[1-9]\d{5}(?!\d)/，中国邮政编码为 6 位数字。

(10) 匹配身份证：/\d{15}|\d{18}/，中国的身份证为 15 位或 18 位，只匹配位数，没有有效性验证。

(11) 匹配 IP 地址：/\d+\.\d+\.\d+\.\d+/，提取 IP 地址时有用。

(12) 匹配特定数字，例如：

```
/^[1-9]\d*$/          //匹配正整数
/^-[1-9]\d*$/         //匹配负整数
/^-?[1-9]\d*$/        //匹配整数
/^[1-9]\d*|0$/        //匹配非负整数(正整数或 0)
/^-[1-9]\d*|0$/       //匹配非正整数(负整数或 0)
/^[1-9]\d*\.\d*|0\.\d*[1-9]\d*$/                    //匹配正浮点数
/^-([1-9]\d*\.\d*|0\.\d*[1-9]\d*)$/                 //匹配负浮点数
/^-?([1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0)$/       //匹配浮点数
/^[1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0$/           //匹配非负浮点数(正浮点数 + 0)
/^(-([1-9]\d*\.\d*|0\.\d*[1-9]\d*))|0?\.0+|0$/      //匹配非正浮点数(负浮点数 + 0)
```

(13) 匹配特定字符串

```
/^[A-Za-z]+$/          //匹配由 26 个英文字母组成的任意长度的字符串
/^[A-Z]+$/             //匹配由 26 个英文字母的大写组成的任意长度的字符串
/^[A-Za-z0-9]+$/       //匹配由数字和 26 个英文字母组成的字符串
/^\w+$/                //匹配由数字、英文字母或者下画线组成的任意长度的字符串
```

5.5.3 数学对象 Math

在 JavaScript 中，Math 对象封装了常用的数学常数和一些常用的数学运算，这些运算包括：三角函数、对数函数、指数函数和一些舍入函数等。

1. Math 对象属性

Math 对象中的属性与其他对象的属性有一些区别。这些属性是常用的数学常数，它们是定值，因此它们是只读的，不允许对这些对象属性进行写操作。表 5-6 列出了 Math 对

象的属性名、描述和近似值。

表 5-6 Math 对象属性

属 性 名	说 明
E	常数 e,也称欧拉常数,它是自然对数的底。近似值为 2.718
LN2	2 的自然对数,即以 e 为底的 2 的对数,近似值为 0.693
LN10	10 的自然对数,近似值为 2.302
LOG2E	以 2 为底的常数 E 的对数,近似值为 1.442
LOG10E	以 10 为底的常数 E 的对数(常用对数),近似值为 0.434
PI	π 常数,即圆周长和直径之比,近似值为 3.14159
SQRT1-2	1/2 的平方根,近似值为 0.707
SQRT2	2 的平方根,近似值为 1.414

2. Math 对象成员方法

Math 对象的成员函数分为三角函数、反三角函数、对数、指数函数、舍入函数以及随机函数等,常用函数见表 5-7。

表 5-7 Math 对象成员方法

成 员 函 数	说 明
sin(x),cos(x),tan(x)	求 x 的正弦、余弦和正切值,x 为弧度值
asin(x),acos(x),atan(x)	求 x 的反正弦、反余弦和反正切值,x 为弧度值
atan2(x,y)	直角坐标系中(x,y)点与 x 轴所成的角度
exp(x)	e 的 x 次方
log(x)	x 的以 e 为底的自然对数
pow(x,y)	计算 x 的 y 次方
sqrt(x)	x 的平方根
abs(x)	x 的绝对值
round(x)	x 四舍五入值
ceil(x)	大于或者等于 x 的最小整数值,向上取整或向上舍入
floor(x)	小于或者等于 x 的最大整数值,向下取整或向下舍入
random()	产生随机数

5.5.4 日期对象 Date

Date(日期)对象封装了有关日期和时间的一些变量和函数,利用这些变量,可以掌握当前日期和时间。Date 对象中没有一个属性是可以直接地设定或者取得的,这种思想更符合对象的封装思想,即成员变量都是私有的,提供公有的成员函数来访问私有的成员变量。

1. 创建日期对象

创建日期对象和数组对象有些相似,都需要使用 new 关键字,但它的构造函数比较复杂,有多个不同参数的构造函数,我们可以根据需要选择一种格式来生成一个新的日期对象,下面分别给出它们的语法和范例:

格式 1: var DateObj = new Date();

这是最简单的一种日期对象的创建格式。创建该对象时,激活默认的构造函数,该构造

函数取计算机的当前时间作为日期对象的时间值。

格式 2：var DateObj = new Date(年,月,日);

这种格式中也有几点要说明：

(1) 参数是数值,不是字符串,年份取后两位。

(2) 当“月”超过 12 或“日”超过当月天数时,将自动进位换算到下一年和月。

(3) “月”参数取值是从 0 到 11,即实际月份比参数大 1。

(4) 小时、分、秒将被认为是 0。

格式 3：var DateObj= new Date(年,月,日,小时,分,秒);

这种格式与前一种很相近,例如：var Date1=new Date(96,3,23,20,55,00);

格式 4：var DateObj = new Date("月日,年小时：分：秒");

在这种格式中,参数是一个字符串,描述了新建 Date 对象的各个属性,该字符串需要满足以下情况：

(1) 在月、日之间的空格可以省略,但是在年、时间之间的空格不可以省略,否则会使日期无效。月份必须用英语的 January,Februlary,…,December,不能用数字。

(2) 日、年之间的逗号不能省略。

(3) 时间部分的小时、分、秒间的冒号不可省略。

(4) 当“日”这一项超过当月的天数时,将自动进位换算到下一个月。“小时”这一项超过 24 时也将进位换算成日。

例如：var meetDate = new Date("June 22,2016 11:50:00");

2. Date 对象常用方法

对于 Date 对象,不同的浏览器支持不完全相同,下面是一组常用的 Date 对象方法,见表 5-8。

表 5-8 Date 对象常用方法

方 法 名	功 能
getDate()	返回 Date 对象一个月中的某一天(1～31)
getDay()	返回 Date 对象一周中的某一天(0～6)
getMonth()	返回 Date 对象的月份(0～11)
getFullYear()	返回 Date 对象以四位数字返回年份
getHours(),getMinutes(),getSeconds()	返回 Date 对象的小时(0～23)、分钟(0～59)和秒数(0～59)
setDate()	设置 Date 对象中月的某一天(1～31)
setMonth()	设置 Date 对象中的月份(0～11)
setFullYear()	设置 Date 对象中的年份(四位数字)
setHours(),setMinutes(),setSeconds()	设置 Date 对象中的小时(0～23)、分钟(0～59)和秒数(0～59)
setTime()	以毫秒设置 Date 对象
toString()	把 Date 对象转换为字符串
toDateString()	把 Date 对象的日期部分转换为字符串
toTimeString()	把 Date 对象的时间部分转换为字符串

除了上述这些常用的方法外,Date 对象还包含许多关于世界时和本地时间的设置方法,在此省略。

【例 5-4】 编写程序,在页面上显示一个走动的时钟,并且当用户关闭页面时,显示该页面停留的时间。

说明:获取页面的停留时间,需要对打开页面和关闭页面时间编程即可。

代码清单如下:

```
<html>
<head>
<meta charset="utf-8">
<script type="text/javascript">
var timeStr="", dateStr="";
var timebegin = new Date();
function timeclock()
{
  now= new Date();
  //时间
  hours= now.getHours();
  minutes= now.getMinutes();
  seconds= now.getSeconds();
  timeStr = ((hours<10)?"0":"") + hours;
  timeStr+= ((minutes<10)?":0":":") + minutes;
  timeStr+= ((seconds<10)?":0":":") + seconds;
  document.myclock.mytime.value = timeStr;
  //日期
  day= now.getDate();
  month= now.getMonth()+1;
  year= now.getFullYear();
  dateStr = year + "/" + month +"/" +day;
  document.myclock.mydate.value = dateStr;
  Timer= window.setTimeout("timeclock()",1000);
}
function timespan()
{
timeend = new Date();
minutes = (timeend.getMinutes() - timebegin.getMinutes());
seconds = (timeend.getSeconds() - timebegin.getSeconds());
time = (seconds + (minutes * 60));
window.alert('您在本页共停留了 ' + time+" 秒钟");
}
</script>
</head>
<body onload=" timeclock()" onunload="timespan();">
  <form name="myclock">
     时间:<input type="text" name="mytime" value="" size="10"><br>
     日期:<input type="text" name="mydate" value="" size="10">
```

```
  </form>
</body>
</html>
```

显示结果如图 5-8 所示。

图 5-8　时钟显示示例

在上述代码中，采用了 UTF-8 编码，因此，如果使用 Windows 记事本编辑上述网页，在保存文件时需要选择编码方式为 UTF-8，否则网页显示为乱码，因为记事本程序文件存储的默认编码为 ANSI 编码。在上述程序中，用到了 window 对象、document 对象和 HTML DOM 对象，详细介绍见后面的讲解。

5.5.5　数组对象 Array

在计算机程序设计语言中，数组是用于保存一组数据的复杂数据类型。因为 JavaScript 为弱类型语言，不可能提供数组类型，数组被定义成一种内置的 Array 对象。用户通过新建 Array 对象来创建数组，并对数组对象进行操作。

在 JavaScript 中，数组的元素不一定是相同的，元素可以是不同类型的。数组的每一个成员对象都有一个“下标”，用来表示它在数组中的位置（下标从 0 开始）。程序通过括号操作符（[]）访问数组成员。

1. 一维数组

在 JavaScript 中，要使用数组，必须创建 Array 对象，创建 Array 对象，有三种形式：

（1）定义空数组

```
var <数组名> = new Array();
```

对于空数组，数组中元素的个数不确定，接下来可以为数组添加元素。

一般形式是：

```
<数组名>[<下标>] = <表达式>;
```

例如：

```
var colorArray = new Array();          //定义一个空数组
```

```
colorArray[0] = "red";
colorArray[1] = "green";
colorArray[2] = "blue";
colorArray[3] = 255;
```


(2) 创建确定元素数量的空数组

```
var <数组名> = new Array(size);
```

参数 size 是期望的数组元素个数,数组对象的 length 属性值将被设为 size 的值。

(3) 创建数组并初始化数组元素

用户还可以在定义数组的时候直接初始化元素数据,一般形式是:

```
var <数组名> = new Array(<元素 1>, <元素 2>, <元素 3>, ...);
```

当使用这些参数来新建数组对象时,构造函数将用参数值作为新建数组的元素值,数组对象的 length 属性也会被设置为参数的个数。

例如,var myArray = new Array(1, 4.5, "Hi ");新建一个数组对象 myArray,数组包含的元素分别是:myArray[0] =1; myArray[1] = 4.5; myArray[2] = "Hi "。

但是,如果元素列表中只有一个元素,而这个元素又是一个正整数的话,这将定义一个包含正整数个空元素的数组。例如,var myArray = new array(10),定义一个长度为 10 的数组,而不是一个数组,包含一个正整数元素 10。

2. 多维数组

JavaScript 只有一维数组,不能和一般的程序设计语言一样,使用类似 var myArray= new Array(3,4)的方法来定义 3×4 的二维数组,或者用 myArray[1,1]来访问"二维数组"中的元素。实际上,所谓多维数组,就是数组的元素本身也是一个数组。因此,要使用多维数组,在 JavaScript 中,可用下面的形式来定义:

var myArray = new Array(new Array(), new Array(), new Array(), ...);

例如:要定义一个 3×4 的二维数组,定义如下:

var myArray = new Array(new Array(4), new Array(4), new Array(4));

然后,可以用 myArray[i][j]的形式访问"二维数组"中的元素。

在 JavasCript 中,数组中的元素可以是不同类型的,因此可以定义 oneArray = new Array(new Array(3), new Array(4)),它不是一个 3×4 的二维数组,实际上 oneArray 有两个元素,第一个元素是一个长度为 3 的数组,第二个元素是一个长度为 4 的数组。

3. Array 对象属性

Array 对象常用的属性是 length 属性,保存数组的长度,即数组里元素的个数,它等于数组里最后一个元素的下标加 1。因此,想添加一个元素,可写做:myArray[myArray.length] = ...。对于二维数组,例如上面的 myArray,document.write(myArray.length)将返回 3,而不是返回 3×4=12。也就是说 myArray 有 3 个元素,都是长度为 4 的数组。

4. Array 对象方法

对于 Array 对象,不同的浏览器支持不完全相同,下面是一组常用的 Array 对象方法,见表 5-9。

表 5-9　Array 对象常用方法

方　法　名	功　　能
slice(<s>[,<e>])	返回一个从第 s 个元素到第 e 个元素的一个子数组，如果不给出 e，则返回的子数组从第 s 个元素到数组的最后一个元素
shift()	删除并返回数组的第一个元素
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多个元素，并返回新的长度
unshift(e1,e2,....)	向数组的开头添加一个或多个元素，并返回新的长度
a. concat(a1,a2,...)	将数组 a1,a2,… 中的元素添加到数组 a 中
reverse()	颠倒数组中元素的顺序
sort()	对数组中的元素进行排序
toString()	把数组转换为字符串，并返回结果
join(<分隔符>)	把数组中的各个元素串起来，用<分隔符>作为元素之间的分隔符，然后返回这个字符串

例如：

```
<script type = "text/javascript">
var aa = new Array(2)
aa[0] = "George"
aa[1] = "John"
var bb = new Array(3)
bb[0] = "James"
bb[1] = "Adrew"
bb[2] = "Martin"
document.write(aa.concat(bb))
</script>
```

在 JavaScript 中，可以直接输出一个数组对象，例如 document. write(aa)，则输出数组中的每一个元素的值，元素值之间用逗号分开。

5.5.6　全局函数

在软件编程时，除了内置对象外，还有一些功能是经常需要使用的，这些功能被定义为标准函数，又称内置函数。内置函数不属于任何对象，因此不用通过引用对象的方式来使用它们，称为 JavaScript 全局函数。常用的全局函数见表 5-10。

表 5-10　JavaScript 全局函数

函　数　名	说　　明
parseInt(str, radix)	解析一个字符串，并返回一个整数。参数 str 为要被解析的字符串，参数 radix 表示要解析的数字的基数，范围为 2～36
parseFloat(str)	解析一个字符串，返回字符串中的第一个数字。如果字符串的第一个字符不能被转换为数字，返回 NaN
Number(object)	把对象值转换为数字。如果参数是 Date 对象，Number()返回从 1970 年 1 月 1 日至今的毫秒数。如果对象的值无法转换为数字，则返回 NaN

续表

函 数 名	说 明
isNaN(x)	检查参数 x 是否是非数字值。如果 x 是非数字值 NaN,返回 true。如果 x 是其他值,则返回 false
eval(str)	将字符串参数 str 看作一个 JavaScript 表达式,并把它作为脚本代码来执行,如果有计算结果,则返回计算结果
String(obj)	把参数 obj 对象转换为字符串。参数 obj 为 JavaScript 对象
isFinite(num)	检查参数 num 是否无穷大。如果是有限数字,返回 true。否则,如果是 NaN(非数字)、正、负无穷大的数,返回 false
encodeURI(str)	把字符串 str 作为 URI 进行编码并返回 URI 编码,其中某些字符被十六进制的转义序列替换。如果 str 中含有分隔符,比如? 和 #,则应当使用 encodeURIComponent()方法分别对各组件进行编码
encodeURIComponent(str)	把字符串 str 作为 URI 组件进行编码。参数 str 含有 URI 组件或其他要编码的文本
decodeURI(URIstr)	对 encodeURI()函数编码过的 URI 进行解码,返回解码后的字符串,其中的十六进制转义序列将被它们表示的字符替换
decodeURIComponent(URIstr)	对 encodeURIComponent()函数编码的 URI 进行解码

【例 5-5】 JavaScript 字符串函数 eval()常常用于动态地得到变量名的操作,阅读下列代码,分析运行结果。

```
<script type = "text/javascript">
var score1 = 97;
var score2 = 100;
for (i = 0;i < 2;i++)
{
   valstr = "score" + (i + 1);                    //构造变量名
   document.write("valstr = " + eval(valstr) + "<br>");
}
</script>
```

上述代码中,声明了两个整数变量,用循环语句输出它们的值,在循环中构造变量名,通过函数 eval(str)来得到一个 JavaScript 代码计算。

5.6 浏览器对象

每一个符合 ECMAScript 规范的脚本程序设计语言都包含三个部分,即 ECMAScript 规范,浏览器对象模型(Browser Object Model,BOM)和文档对象模型(Document Object Model,DOM)。所谓浏览器对象模型 BOM,就是指当用户打开浏览器时,浏览器中的 JavaScript 运行时引擎将在内存中自动创建一组对象,用于对浏览器及 HTML 文档对象模型中数据的访问和操作。因为这些对象是和浏览器本身紧密相关的,故称为浏览器对象。

5.6.1 浏览器对象模型 BOM

当使用浏览器打开一个网页时,浏览器就在内存中创建了一个窗口对象(window),它

封装了浏览器的整个窗口，如果文档中包含框架(frame 或 iframe 元素)，浏览器还会为每个框架创建一个子 window 对象。在一个窗口中，包含了窗口标题、地址栏、客户区、状态栏等，它们也被定义为独立的浏览器对象，都是 window 对象的成员对象，这些对象构成一种层次结构，即浏览器对象模型。

在 JavaScript 中，根据窗口的构成，定义的浏览器对象及层次结构如图 5-8 所示。

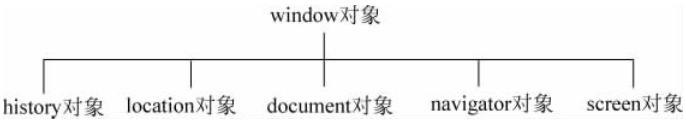

图 5-8　浏览器对象模型 BOM 层次结构

在浏览器对象模型的 7 个对象中，window 对象是最顶层的对象，它对应了浏览器窗口本身，其他 6 个对象都是 window 对象的成员对象(对象名的首字母均为小写字母)，其中 document 成员对象也是 HTML 文档对象模型 DOM 中的重要对象。浏览器对象模型中的每一个对象都是预声明的内存对象，用户可以直接使用。

5.6.2　窗口对象 window

当浏览器打开一个网页文件，如果新建了一个浏览器窗口，则在计算机内存中创建一个窗口对象，对象名为 window。在客户端，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，要引用当前窗口不需要指定窗口对象本身，而是可以把 window 对象的属性作为全局变量来使用。例如，可以将 window. document 简写为 document，window. alert()可以简写为 alert()一样。这是对窗口对象的隐式应用，如果要显式地表明窗口，可写为 window.[属性|方法]或 self. [属性|方法]。

一个 window 对象封装了浏览器窗口的全部内容，从浏览器窗口可以很容易地理解 window 对象包含的属性和方法。例如：一个浏览器窗口都包括：标题栏、前进后退按钮、地址栏、客户区、滚动条、状态栏等。这些窗口元素构成了 window 对象的成员，为操作方便，许多成员本身又被封装为对象，构成 window 对象的成员对象。

1. window 对象常用属性

window 对象的属性可分为一般属性和对象属性，对于每一个属性，具有不同的读写权限，即有的属性为只读属性，不能为属性进行赋值，有的属性则可以进行读写操作。window 对象一般属性及成员对象见表 5-11。

表 5-11　window 对象属性表

属　性　名	说　　明
name	窗口名称，窗口名称可以用作一个< a >或者< form >标记的 target 属性值
window，self	指向当前窗口，相当于 this
history	封装浏览器窗口中“前进”“后退”按钮，包含用户在当前浏览器窗口中访问过的 URL
location	封装浏览器窗口的地址栏，将一个网址赋给 location 属性，该网页将在浏览器窗口打开
document	封装浏览器窗口打开的 HTML 文档，是 DOM 模型的根

续表

属 性 名	说 明
defaultStatus status	封装浏览器窗口状态栏,属性值可读写
frames	如果 window 中包含帧,则 frames 为一个数组对象,保存每个帧对应的子窗口。可以通过 window.frames["frame-name"]或数组下标 0..n 来访问子窗口对象
opener	创建此窗口的父窗口引用
top	顶级窗口指针
closed	窗口是否关闭标志。当浏览器窗口关闭时,该窗口对应的 windows 对象并不会消失,它的 closed 属性为 true
innerHeight innerWidth	窗口中文档显示区域的高度和宽度,不包括菜单栏、工具栏等部分。IE 对应的属性是 body 对象的 clientHeight、clientWidth 属性
screenTop screenLeft	窗口距离屏幕顶部和左侧的距离,即窗口左上角的 x、y 坐标

【例 5-6】 编写一个网页文件 1.htm,确保不被嵌入到一个框架中打开。

分析:利用 window 对象的 self 和 top 属性可以判断窗口是否在一个框架中,如果是,则跳出框架,代码如下:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<script type="text/javascript">
  if (window.top!= window.self) {
      window.top.location="1.htm"
  }
</script>
</body>
</html>
```

2. window 对象常用方法

浏览器对象 window 的常用方法见表 5-12。

表 5-12 window 对象常用方法

方 法	说 明
open(URL,Name,Features)	打开一个新浏览器窗口,并返回该窗口对象。 URL,可选参数,新窗口中要显示的文档的 URL,取值为空时,新窗口不显示任何文档。 Name,可选参数,新窗口名称。如果该参数指定了一个已经存在的窗口,那么 open()方法不再创建一个新窗口,而返回对指定窗口的引用。 Features,可选参数,设置窗口特征
close()	关闭浏览器窗口,只有通过 JavaScript 代码打开的窗口才能够由 JavaScript 代码关闭,这阻止了恶意脚本终止用户的浏览器
focus()	将键盘焦点设置到一个窗口
blur()	把键盘焦点从顶层浏览器窗口移走,整个窗口由 Window 对象指定。哪个窗口最终获得键盘焦点并没有指定

续表

方　法	说　明
moveTo(x,y)	把窗口的左上角移动到一个指定的坐标(x,y)处
moveBy(x,y)	相对窗口的当前坐标把它在 x,y 方向移动指定的像素
resizeTo(width,height)	将窗口的宽度和高度调整为 width 和 height 个像素
resizeBy(width,height)	将窗口的宽度和高度增大或减少 width 和 height 个像素
scrollTo(x,y)	将浏览器窗口左上角定位到文档区域的(x,y)坐标位置,文档上边和左边的部分将滚动出浏览器窗口
scrollBy(xnum,ynum) scroll(xnum,ynum)	将浏览器窗口左上角分别向右、向下滚动 xnum 和 ynum 像素
setInterval(code,millisec)	按指定周期(毫秒)调用函数或计算表达式。 code,必选参数,要调用的函数或要执行的代码串。 millisec,周期性执行或调用 code 函数的时间间隔,以毫秒计
setTimeout()	在指定的毫秒数后调用函数或计算表达式。参数同 setInterval(),setTimeout()只执行 code 一次。如果要多次调用,请使用 setInterval()或者让 code 自身再次调用 setTimeout()
clearInterval(id)	取消由 setInterval()设置的 timeout。参数 id 必须是由 setInterval()返回的 ID 值
clearTimeout(id)	可取消由 setTimeout()方法设置的 timeout
alert(message)	显示带有一条指定消息和一个“确定”按钮的警告框
confirm(message)	显示一个带有指定消息和“确定”“取消”按钮的对话框。如果用户单击“确定”按钮,则 confirm()返回 true,否则返回 false
print()	打印当前窗口的内容。类似用户单击浏览器的“打印”按钮

在 window 对象的方法中,方法 scrollTo()和 scrollBy()比较难理解。我们说 scrollTo(x,y)中(x,y)是浏览器窗口左上角在文档区域中的(x,y)坐标。这听起来比较拗口,其实文档和观察文档的浏览器窗口的移动是相对的。当我们拖动滚动条的时候,好像窗口不动,文档在动。如果说文档不动,窗口在动,其实结果是一样的。后一种情况虽然难理解,但更加符合实际。

通常情况下,文档的尺寸(宽度和高度)随着文档内容的多少而变化。但是,我们也可以设置一个很大的文档区,例如:body{height: 10000px;width: 10000px;},文档的实际内容其实没有这么宽和这么高,它位于 10000×10000 区域的左上部。通过 scrollTo(x,y)可以将浏览器窗口左上角定位到 10000×10000 区域的某个位置,这和拖动滚动条的效果是一样的。

对于 setInterval()和 setTimeout()两种方法,前者定义周期性执行的函数更加方便,不需要递归调用接口实现。例如,下列脚本程序可以在页面中显示一个走动的时钟:

```
<script type = "text/javascript">
var clockid = self.setInterval("clock()",1000);
function clock()
{
  var nowtime = new Date();
  form1.clock.value = nowtime.toTimeString().substring(0,8);
}
</script>
```

当打开一个窗口时,可设置的窗口特征见表 5-13。

表 5-13 窗口特征列表

特 征	说 明
top,left	窗口左上角坐标的像素值
height,width	窗口的高度和宽度
titlebar	是否显示标题栏,取值为 yes|no|1|0。默认值为 yes
menubar	是否显示菜单栏,取值为 yes|no|1|0。默认值为 yes
toolbar	是否显示浏览器的工具栏,取值为 yes|no|1|0。默认值为 yes
location	是否显示地址字段,取值为 yes|no|1|0。默认值为 yes
status	是否添加状态栏,取值为 yes|no|1|0。默认值为 yes
scrollbars	是否显示滚动条,取值为 yes|no|1|0。默认值为 yes
resizable	窗口是否可调节尺寸,取值为 yes|no|1|0。默认值为 yes
fullscreen	是否使用全屏模式显示浏览器。默认值为 no。处于全屏模式的窗口必须同时处于剧院模式

【例 5-8】 当在浏览器中打开一个页面时,如果文档长度大于浏览器窗口的高度,则出现垂直滚动条,编写代码,使得窗口滚动条置于窗口的底部。

分析:这是在实际项目研发中常用的功能,它可能是单击上部一个按钮,在文档尾部添加了一条记录,应该自动地定位到文档底部。

代码清单:

```
<html>
<head>
<meta charset="utf-8">
<style>
#div1{
  position:absolute;
  top:10px;
  left:10px;
  width:100%;
  background-color:#ff0000;
  border:1px solid #00ff00;
}
#div2{
  width:70%;
  height:100%;
  overflow:auto;
}
</style>
<script type=text/javascript>
function gobottom()
{
  //将窗口的滚动条定位到页面底部,本处未用到
  var c = window.document.body.scrollHeight;
  window.scrollBy(0,c);
  //将 div 的滚动条移动到底部
  var divobj = document.getElementById('div2');
  divobj.scrollTop = divobj.scrollHeight;
```

```
}
function gotop()
{
  window.scrollTo(0,0);
  document.getElementById('div2').scrollTop = 0;
}
</script>
</head>
<body>
<div id = "div1">
<input type = "button" name = "b1" value = "滚动条置底" onclick = "gobottom();">
</div>
<div id = "div2">
<script type = "text/javascript">
for (i = 1;i < 50;i++)
    window.document.write("Line" + i + "<br>");
</script>
<a href = "#" onclick = "gotop();return false;">回顶部</a>
</div>
</body>
</html>
```

在浏览器中打开上述页面，显示结果如图 5-9 所示。

图 5-9　滚动条置底操作页面

在上述代码中，有三个在实际项目研发中遇到的重要的技术难点，解释如下：

(1) 滚动条有两种情况，一种是窗口滚动条，一种是 div 滚动条。在图 5-9 中我们看到滚动条是 div2 的滚动条。为什么窗口没有出现滚动条呢？这是与 div 的样式定义中设置了 height:100%有关。如果删除 div 的 height 属性设置，则会出现窗口滚动条，而 div2 不出现滚动条。因为，当设置了 div2 的 height 属性后，它不会超过窗口的高度，因此，窗口不会出现滚动条。但是，如果没有设置高度，则 div2 的高度是变化的，当内容超过窗口高度时，窗口自然会出现滚动条，但 div2 不会出现滚动条，因为 div2 并未设置高度。

(2) 滚动条置底，通常情况下，滚动条会出现在窗口和 div 中，在窗口中，可以比较容易地控制滚动条的位置，即通过 gobottom()函数中的 window.scrollBy(0,c)即可实现。

(3) 将 div 块固定,不随滚动条的滚动而滚动,且不闪烁。在通常情况下,当拖动窗口滚动条时,div 块也会随着滚动条的滚动而滚动。如何让一个 div 块不会随滚动条而滚动呢? 方法就是将< body >分成两个 div,一个输出固定不动的内容,另一个是其他内容。如上面代码中的 div1 和 div2。然后,设置 div2 块的 height:100%,这样窗口就不会出现垂直滚动条了。

当 div2 中的输出内容超过 div2 设置的高度时,则< div >出现垂直滚动条,看上去类似窗口的滚动条,移动 div2 的滚动条,当然不会影响 div1,这样就看到 div1 是固定不动的。此外,还可以将 div 的 z-index 设置为-1,使得在有多个图层叠加时,该 div 处于底层。处于底层的 div 中的输入元素不能获得输入焦点。

5.6.3 地址栏对象 location

所有 Web 浏览器都包含地址栏,这是用户输入网址的地方。在 JavaScript 中,浏览器窗口的地址栏被封装成 location 对象,它是 window 对象的成员对象。地址栏对象 location 封装了浏览器窗口的地址栏,其常用属性和方法见表 5-14。

表 5-14 location 对象常用属性

属 性 名	说 明
href	存储地址栏网址 URL,可以通过为该属性设置新的 URL,浏览器将打开新的网页
protocol	存储 URL 中的协议部分,包括后面的冒号(:)
hostname	存储 URL 中的主机名+域名部分,不包括端口号
host	存储 URL 的主机名和端口号,只有端口号是 URL 的一个明确部分时,值中才包括端口号
port	存储 URL 的端口部分
pathname	存储 URL 中文件路径。如果 URL 中的网页文件是根下的一个文件,则 location. pathname 的值为根(/)
hash	该字符串是 URL 的书签部分(从#号开始的部分)
search	存储 URL 的中问号("?")之后的部分,即参数部分

在上述属性中,可以看出 location 对象属性可以分别获取一个 URL 的各个部分,这些属性都可以读写,从而可以在当前窗口打开新的网页。

Location 对象方法见表 5-15。

表 5-15 location 对象常用方法

方 法	说 明
assign(URL)	把一个新的 URL 赋给当前窗口的 location 对象,即在当前窗口打开一个新的网页。也可以通过为 location. href 赋值来导航到一个新的网页,采用 assign 的方法会使代码易维护
reload([true|false])	如果参数设为 true,则从服务器上重新下载网页,等价于单击浏览器刷新(refresh)按钮 如果参数是 false,检测服务器上的文档是否已改变,如果文档已改变,则再次下载该文档。如果文档未改变,则该方法将从缓存中装载文档
replace(newURL)	用一个新文档取代当前文档。不会在 history 对象中生成一个新的记录,新的 URL 将覆盖 istory 对象中的当前记录

例如：下面代码定义一个超链接，单击，则在当前窗口打开一个新的网页。

```
<a href = "#" onclick = "window.location.assign('http://www.google.com')"> Google 搜索</a>
```

【例 5-9】 利用 location 对象，在 htm 页面中，编写一个函数获取并输出传入的参数名及参数值。

分析：在 HTML 中，当打开一个网页时，经常会传入相应的参数，如果调用的是服务器页面(如 JSP 页面)，在服务器页面可以通过服务端的 request.getParameter()；程序来获取页面中的参数，这和表单数据的获取相同。如果调用的是 htm 文档，则需要通过客户端的 location 对象来获取，并且需要使用类。

假设有两个页面 1.htm 和 2.htm，在 1.htm 中，定义了一个超链接：

```
<a href = "2.html?username = jane&age = 18" target = "_blank"> test parameter </a>
```

在网页 2.htm 中，如何获取传入的参数呢？因为 2.htm 不是一个服务器页，无法像服务器页那样，来获取客户端中的表单数据和 URL 参数。在客户端要获取传入的参数，需要在 2.htm 中定义一个求传入参数的类，来解析 URL 中的参数表，然后将解析后的参数名称和参数值，在当前页面创建为内存变量。

页面 2.html 代码清单：

```
<!DOCTYPE html>
<html>
<head>
<meta charset = "utf-8">
<script type = "text/javascript">
function GetParaString()
{
  var pname,pvalue,i = 0;
  var str = window.location.href;                    //获取浏览器地址栏 URL 串
  var num = str.indexOf("?")
  str = str.substr(num + 1);                         //截取"?"后面的参数串
  var parray = str.split("&");                       //将各参数分离形成参数数组
  for (i = 0;i < parray.length;i++)
  {
     num = parray[i].indexOf(" = ");
     if (num > 0) {
        pname = parray[i].substring(0,num);          //取参数名
        pvalue = parray[i].substr(num + 1);          //取参数值
        this[pname] = pvalue;                        //定义对象属性并初始化
     }
  }
}

function showPara(obj)
{
  for(var prop in obj)
     document.write(prop + ":" + obj[prop] + "<br>");
```

```
}
</script>
</head>
<body>
<p>传入参数为:</p>
<script type="text/javascript">
  var Request = new GetParaString();          //创建参数对象实例
  var myname = Request["username"];           //取传入的参数 username
  var myage = Request["age"];                 //取传入的参数 age
  showPara(Request);
</script>
</body>
</html>
```

上述代码完成后,进行测试:在 1.htm 中,单击超链接“test parameter”,则打开一个新窗口,显示 2.html,在页面的顶部,显示传入的参数名和参数值。

5.6.4 显示屏对象 screen

在客户端编程中,常常需要获取用户计算机显示器的有关信息。在 JavaScript 中,用户计算机显示屏的信息被封装在 screen 对象中。JavaScript 程序可以利用这些信息来优化输出,以达到显示要求。Screen 对象常用属性见表 5-17

表 5-17 Screen 对象属性

属 性	描 述
availHeight,availWidth	存储显示器屏幕可用的高度和宽度的像素数
height,width	存储显示器屏幕的高度和宽度的像素数

除了上述属性外,screen 对象还有一组与分辨率、色彩有关的属性,在此省略。此外,和其他对象不同,screen 对象没有提供方法。

由于现在的大多数 Web 浏览器都采用标签式页面管理,这些页面窗口都被组织在浏览器窗口中,这使得网页的切换和关闭变得容易。在标签式页面组织模式下,如果在一个窗口通过 window.open()方法动态地打开一个新窗口,不管如何设置新建窗口的属性,或者移动新窗口的位置,以及对窗口放大缩小等操作,将都不生效,而是和其他窗口一样作为一个页面标签,出现在浏览器窗口中。

5.6.5 浏览器对象 navigator

随着浏览器产品的增多,网页在不同浏览器中可能会显示不同。因为不同的浏览器对 HTML 和 JavaScript 的支持不完全相同。为了保证设计的网页在大多数主流浏览器中都能够正确地显示,不仅要使用通用和更加规范的 JavaScript 代码,常常还需要获取用户打开网页时使用的浏览器信息。

在 JavaScript 的 BOM 中,客户浏览器信息被封装为一个浏览器对象,即 navigator 对象,它封装了有关操作系统、浏览器版本等环境信息,navigator 对象常用属性及方法见表 5-18。

表 5-18　navigator 对象属性及方法

属　　性	
plugins[]	数组成员，存储浏览器已经安装的插件
appName	存储浏览器名称字符串
appVersion	存储浏览器版本号和操作系统信息，不同的浏览器显示的内容项目不同
appMinorVersion	存储浏览器的次级版本
appCodeName	浏览器代码名称
platform	存储客户端操作系统平台
方　　法	
javaEnabled()	检查浏览器是否支持并启用了 Java。如果是，返回 true，否则返回 false

5.7　HTML 文档对象

当浏览器打开一个网页时，不管是 HTML 还是 XML 文档，浏览器在显示文档的同时，浏览器中的 JavaScript 运行时引擎同时还为每一个元素在内存中创建一个内存对象，称为文档对象。在 JavaScript 中，文档对象构成了网页的编程接口，通过对这些可访问的内存对象进行编程，从而实现对网页中元素及其属性的访问和修改，增强网页的交互功能。

5.7.1　文档对象模型 DOM

文档对象(Document Object)是浏览器在打开网页的过程中，对每一个元素在内存中创建的对象，它封装了网页元素的属性和方法。对应网页元素的层次结构，文档对象也形成一种对应的层次关系，以树状结构组织，构成一棵文档树。

为了更好地规范文档对象编程，W3C 发布了文档对象模型(Document Object Model，DOM)规范，以解决不同的浏览器厂商在脚本语言实现中的冲突和标准化问题。DOM 为 Web 应用的前端开发提供了一套标准方法，遵循 DOM 规范，客户端脚本程序将具有更好的兼容性，从而保证网页在不同浏览器中的显示更加一致。

W3C 将 DOM 分为三个不同的部分，即核心 DOM、XML DOM 和 HTML DOM。核心 DOM 是用于任何结构化文档的标准模型，XML DOM 和 HTML DOM 分别是用于 XML 文档和 HTML 文档的标准模型。在 DOM 中，定义了所有文档元素的对象和属性，以及访问它们的方法。

对于 DOM 对象的访问，可以出现在页面的脚本程序中，也可以直接在浏览器地址栏中书写。例如：当浏览网页时，有时候需要知道网页的发布时间，从而判断网页内容是否是很久以前的，此时，可以在浏览器的地址栏中输入：

```
javascript:document.write(document.lastModified)
```

输入结束后，按回车键，则打开一个新的网页显示当前正在浏览的网页的最后修改日期。然后单击浏览器工具栏中的后退按钮，返回到刚才浏览的网页。

1. HTML DOM 对象

当浏览器打开一个 HTML 文档时,对应文档中的每一个元素,在内存中创建一个文档对象,文档对象的属性和方法对应了元素的一般属性和事件属性。对应元素的层次关系,这些文档对象表现为层次结构,构成一棵 HTML 文档对象树。

根据 HTML DOM 的概念,每一个 DOM 对象都对应一个 HTML 元素,因此,网页中有什么元素就有什么 DOM 对象,常见的 HTML DOM 对象见表 5-19。

表 5-19 常用 HTML DOM 对象

DOM 对象	说 明	DOM 对象	说 明
document	封装整个 HTML 文档,可用来访问页面中的所有元素	Form	封装< form >元素
meta	封装一个< meta >元素	input text	封装表单中的一个文本框
link	封装一个< link >元素	input password	封装表单中的一个密码域
style	封装一个单独的样式声明	textarea	封装< textarea >元素
body	封装< body >元素	button	封装< button >元素
Image	封装< img >元素	input radio	封装表单中的一个单选按钮
Anchor	封装< a >元素	input checkbox	封装表单中的一个复选框
Table	封装< table >元素	select	封装表单中的一个选择列表
TableRow	封装< tr >元素	option	封装< option >元素
TableData	封装< td >元素	input file	封装表单中的一个文件上传
frameset	封装< frameset >元素	input hidden	封装表单中的一个隐藏域
frame	封装< frame >元素	Event	封装某个事件的状态
iframe	封装< iframe >元素		

在上述表格中,在 DOM 对象列,分为两种情况:①对象,例如 document、body 对象,因为一个 HTML 文档只有一个 document 和一个 body 对象。②对象类,例如 Image,因为一个 HTML 文档可能包含多个< img >标记,每个标记都创建一个 Image 对象。在所有的 HTML DOM 对象中,它们之间为层次结构关系,这种关系和文档中元素的层次关系一致。因此,可以说所有元素对象都是 document 对象的成员对象。同时,文档又是浏览器窗口的一部分,因此,document 对象又是 window 对象的成员对象。HTML DOM 对象层次结构关系如图 5-10 所示。

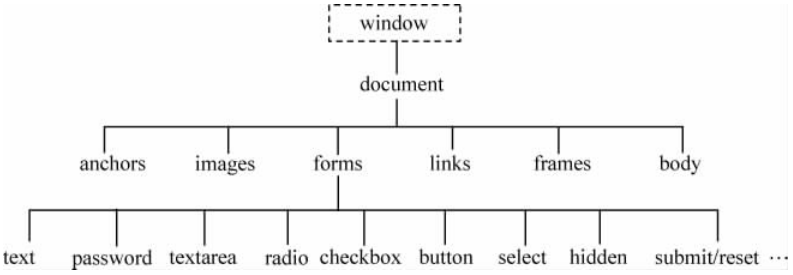

图 5-10 HTML DOM 对象层次结构

2. DOM 对象属性及方法

在计算机软件中,一个对象包括属性和方法两部分。在 HTML DOM 中,每一个文档

对象都对应一个 HTML 元素，是对应 HTML 元素的封装。HTML 元素的一般属性对应 DOM 对象的属性，HTML 元素的事件属性对应 DOM 对象的方法。此外，用户也可以自定义 HTML 元素属性，这些属性也一并成为对应 DOM 对象的属性和方法。因此，当我们熟悉了 HTML 标记的属性后，HTML DOM 对象及其属性和方法是很容易理解的。

例如：对应< img src= "smile.jpg" style="position:absolute; bottom:0px">，将创建一个 Image 对象，该对象必然有一个 src 属性和 style 属性，且 src 属性的值为“smile.jpg”，style 属性将图片定位在窗口的底部。通过 JavaScript 编程，可以设置 Image 对象新的属性值，从而对网页中< img >标记的图片进行操作，例如：改换图片文件，改变图片位置等。

5.7.2 文档对象 document

当浏览器打开一个 HTML 网页文件时，在内存中创建一个文档对象，对象名为 document。文档对象 document 封装了整个 HTML 文档，因此，在 HTML DOM 层次结构中，所有元素对象都可看作是 document 对象的成员对象。

1. document 对象属性

文档对象 document 封装了整个 HTML 文档，因为一个 HTML 文档是由文本、图片、超链接、表格、表单等元素构成的，这些元素在内存中也会创建相应的对象，这些对象构成了 document 对象的属性，document 对象常见属性见表 5-20。

表 5-20 document 对象常见属性列表

属 性 名	说 明
all[]	成员数组，存储文档中所有 HTML 元素对象
images[]	成员数组，存储文档中所有< img >元素对应的 Image 对象
anchors[]	成员数组，存储所有< a name="bookname">元素对应的 Anchor 对象，如果含有 href 属性，则对象保存到 links[]数组中
links[]	成员数组，存储所有< a href="url">超链接元素对应的 Link 对象
forms[]	成员数组，存储文档中所有< form >元素对应的 Form 对象
body	成员对象，对应文档< body >元素。因为一个 HTML 只有一个< body >，因此，对文档体内元素的访问，可直接通过 document 对象进行
lastModified	存储文档最后修改的日期和时间
cookie	存储当前文档有关的所有 cookie
domain	存储当前文档的域名
URL	存储文档的 URL
referrer	存储载入当前文档的文档 URL

文档对象 document 包含的属性很多，这与 HTML 文档中包含的元素有关。但是不是所有的文档元素都被封装成一个 document 的成员，例如 table 元素。理论上讲，document 对象有两个成员对象，即 head 元素和 body 元素，其他文档元素都是 body 对象的成员。但是，由于一个 HTML 文档只有一个 body，为操作方便，对文档中元素的访问通常可以直接通过 document 完成，而不是写成“document.body.元素”的形式。

2. document 对象方法

文档对象 document 常用方法见表 5-21。

表 5-21 document 对象常用方法列表

方 法	功 能
open(mimetype,replace)	打开一个输出流,返回一个新的 document 对象。 参数 mimetype,可选,设置文档类型,默认值为"text/html"。 参数 replace,新文档将覆盖当前网页内容
write(exp1,exp2,...)	在文档输出,如果是一个逻辑表达式,则输出 true 或 false,如果是一个 Array 对象,则输出数组元素的值,多个元素值之间用逗号分开。如果是对象,则输出属性名和属性值
writeln(exp1,exp2,...)	功能同 write()方法,输出结束后输出一个换行符
close()	关闭用 open()方法打开的输出流
getElementById(id)	返回文档中元素 id 属性为 id 的 DOM 对象。一个结构良好的 HTML 文档,元素的 id 属性是唯一的
getElementsByName(name)	返回文档中元素 name 属性为实际参数 name 的对象的集合。因为一个文档中元素的 name 属性可能不唯一,所以该方法可能返回一个简单变量,也可能返回一个数组
getElementsByTagName(tagname)	返回文档中指定标签名的对象集合。返回元素的顺序是它们在文档中的顺序。如果文档中包含多个相同的标记,则返回一个数组

在 HTML DOM 中,定义了多种访问 DOM 对象的方法,除了 getElementById()之外,还有 getElementsByName()和 getElementsByTagName()。如果需要查找文档中的一个特定的元素,最有效的方法是 getElementById()。在操作文档的一个特定的元素时,最好给该元素一个 id 属性,为它指定一个(在文档中)唯一的标识,然后就可以用该 ID 访问该元素。可以用 getElementsByTagName()方法获取任何类型的 HTML 元素的列表。例如,下面的代码可获取文档中所有的表:

```
var tables = document.getElementsByTagName("table");
window.alert ("This document contains " + tables.length + " tables");
```

需要注意的是,getElementsByName()方法和 getElementsByTagName()可能返回一个简单变量,也可能返回一个数组,在脚本编程时,必须要考虑到这两种不同的情况,否则将会出现 JavaScript 脚本运行错误。因为,编程时,对简单变量和数组的访问是不一样的。

3. 访问 HTML DOM 对象方法

利用文档对象 document,我们可以用多种方法获得一个 HTML 元素对应的 DOM 对象,主要可以有以下几种方法:

(1) 利用标记的 id 属性访问对象

在同一个 HTML 文档中,标记的 id 属性应该是唯一的,因此可以通过 id 来访问对应的内存 DOM 对象。有两种用法:

方法 1:通过元素 id 属性获得对应的 DOM 对象

```
var obj = document.getElementById("element-id");
```

方法 2:直接使用 id 标识

```
element-id.[属性|方法] = ...
```

如果有一个标识元素 id 的字符串，使用 eval()函数，可以得到对应的 DOM 对象，即：

```
obj = eval("element - id")
```

理论上讲，上述三种方法都可以获取 DOM 对象，实现对 HTML 元素的访问和操作。但是，在浏览器中，对上述方法的支持并不一致。第一种方法是所有浏览器都支持的。

(2) 利用标记的 name 属性访问对象

在 HTML 文档中，如果一个元素声明了 name 属性，JavaScript 运行时引擎在创建 DOM 对象时，元素的 name 属性即成为对应 DOM 对象的名字。通过 name 属性访问对应的内存对象，有两种方法：

方法 1：获取 name 对应的 DOM 对象

```
var obj = document.getElementsByName("element - name");
```

因为在一个 HTML 文档中，元素的 name 属性可以重名，因此返回的可能是一个数组，也可能是一个简单变量。

方法 2：直接使用元素的 name 属性值

对于在< form >中的输入域，通常会给定一个 name 属性，此时可以使用下列形式访问表单中的这些输入域对象：

```
obj = document.form - name.element - name
```

也可以使用

```
obj = document.all.element - name 或 obj = document.all("element - name")
```

对于< form >外的元素，不能通过元素的 name 属性直接访问对象，例如：对于一个< span name="sp1">，则下列两种写法都是错误的：

```
document.sp1.innerText = "xxx",
document.all.sp1.innerText = "xxx"
```

要使用上述写法，需将< span >的 name 属性的设置改为 id 属性的设置，即< span id="sp1">。

(3) 利用 document 对象的成员对象 all 访问

在文档对象 document 中，包含一个数组成员 all，它存储了文档中所有元素对应的 DOM 对象。为增强代码的可读性，这些对象也分类保存在 images、links、anchors 和 forms 数组成员中，利用这几个成员可以访问 DOM 对象。

在 JavaScript 中，数组元素的访问有多种形式，既可以使用数组常用的括号([])操作符，也可以使用对象元素常用的点操作符(.)，还可以使用圆括号操作符。使用 all 成员对象访问 DOM 对象的一般形式有：

```
obj = document.all.element - id
obj = document.all["element - id"]
obj = document.all("element - id")
obj = document.all.item("element - id")
```

需要注意的是，必须使用元素的 id 属性，不能使用 name 属性，因为在一个 HTML 文档

中,name 属性是可以重名的,重名的时候,JavaScript 会创建一个数组,而 id 属性是不能重名的,它唯一地标识了一个元素。

【例 5-10】 正确访问 HTML DOM 对象是进行 JavaScript 编程的基础,阅读下列代码,分析运行结果。

代码清单:accessdoms.htm

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
span{font-size:16px;color:#0000FF;cursor:pointer;}
div {
  position:absolute;
  top:50px;
  left:10px;
  width:150px;
  height:100px;
  border:1px solid #ff0000;
  text-align:center;
  line-height:600%;
}
</style>
<script type="text/javascript">
////////////////////////////////////////////////////////////////////////////////
//参数 obj 为对象
function fun1(obj)
{
  document.form1.t1.value=obj.innerHTML;
}
////////////////////////////////////////////////////////////////////////////////
//参数 idstr 为元素 id 字符串
function fun2(idstr)
{
   obj=document.getElementById(idstr);
   obj.style.bottom="100px";
   //eval(idstr).style.bottom="100px";
}
////////////////////////////////////////////////////////////////////////////////
//无参函数,通过元素 id 直接访问元素对象
function fun3()
{
  winwidth =window.innerWidth?window.innerWidth:document.body.clientWidth;
  winheight=window.innerHeight?window.innerHeight:document.body.clientHeight;
  left=(winwidth-150)/2 + "px";
  if (popupdiv.style.left!=left)
```

```
    {
        popupdiv.style.left = (winwidth - 150)/2 + "px";
        popupdiv.style.top = (winheight - 100)/2 + "px";
    }
    else
    {
        if (popupdiv.style.display == "none")
            popupdiv.style.display = "block";
        else
            popupdiv.style.display = "none";
    }
}
</script>
</head>
<body>
<span onclick = "fun1(this)">参数传递,实际参数为对象</span>||
<span onclick = "fun2('mydog')">参数传递,实际参数为字符串</span>||
<span onclick = "fun3()">居中显示,或隐藏</span>
<img src = "images/dog.jpg" id = "mydog" style = "position:absolute;bottom:0;left:0">
<div id = "popupdiv">
图层在客户区居中
</div>
<form name = "form1">
<input type = "text" name = "t1" value = "">
</form>
</body>
</html>
```

对于上述代码清单,说明如下:

(1) 不同的 HTML 版本,有些代码可能不兼容,例如上述代码中修改对象的 style 属性语句 obj.style.bottom=100,在 HTML 4.01 中是可以的,但在 HTML 5 中,所有关于位置量的都需要给定单位,即 obj.style.bottom="100px"。

(2) 不同的浏览器支持的 DOM 对象属性不完全一样,例如,在函数 fun3()中,求浏览器窗口客户区的高度和宽度,考虑到了 IE 浏览器和其他浏览器的情况。

(3) 对于 JavaScript 函数定义和调用,形式参数可以是字符串,也可以是一个对象,两者在函数内部使用时不同。例如:函数 fun1(obj)形式参数为对象,则在函数调用时,实际参数必须是 DOM 对象,参数 this 为当前对象,在函数内部可以直接对对象操作。函数 fun2(idstr)定义的形式参数类型为代表一个 DOM 对象的 id 字符串,在函数调用时,实际参数为字符串,因此在函数内部,通过 eval(),将参数字符串转化成对应的对象。

在定义 JavaScript 函数时,如果某个参数为标记的 id,在函数调用时,如果实际参数只写 id 号,则参数代表的是一个对象,如果参数写成 id 号两侧用双引号或单引号括起来,则参数为字符串,两者有着很大的不同,直接影响函数内部的语句写法。

在 Google Chrome 浏览器中打开该网页,显示结果如图 5-11 所示。

对于 document 对象,在执行输出操作后,最后一定要使用 close()方法关闭输出流,否

图 5-11　对象访问示例页面显示结果

则,在浏览器的状态栏将显示蓝色进度条,显示下载未完成。例如,在一个数据记录内容修改页面,向一个 iframe(name= filelistbox)中输出内容,代码如下:

```
document.filelistbox.document.write("<body style='margin-top:0px;margin-left:10px;font-size:12px;line-height:150%'>");
document.filelistbox.document.write("<span style='display:none;'>" + iframestr + "</span>");
document.filelistbox.document.write("...");
...
document.filelistbox.document.write("</body>");
document.filelistbox.document.close();
```

5.7.3　文档体对象 body

每一个 HTML 网页文件都有一个< body >标记,对应 body 元素,浏览器在内存中创建一个 body 对象,body 对象封装了 HTML 文件中< body >标记的属性以及所有的文档体中包含的元素对象。从 HTML 结构看,body 对象是 document 对象的成员对象,body 对象常用属性和方法见表 5-22。

表 5-22　document.body 对象属性列表

属　性	说　明
className	存储 body 元素 class 属性值
scrollWidth scrollHeight	存储文档实际的宽度和高度。若宽度大于浏览器窗口的宽度,则显示水平滚动条,若高度大于浏览器窗口的高度,则显示垂直滚动条
topMarign,bottomMargin leftMarign,rightMargin	存储浏览器窗口客户区内容和上、下、左、右边框的距离(像素)
clientWidth clientHeight	存储浏览器窗口客户区宽度和高度
clientLeft clientTop	存储浏览器窗口客户区左上角(x,y)坐标
innerHTML	存储< body >体内的 HTML 内容,包括标记
innerText	存储< body >体内的文本内容,标记被作为文本处理
bgColor	存储< body >背景色
disabled	如果该成员变量设为 true,则页面的内容被灰化,且不可被选取,右击时快捷菜单被禁用

在< body >标记属性中，scrollWidth 和 scrollHeight 存储了文档的宽度和高度。通常情况下，文档的大小随着内容的多少而变化。其实，body 和 div 一样，也是可以设置大小的，例如：body{width:10000px;height:10000px}。文档是通过浏览器窗口来观察的，通过 window 对象的 scrollTo()和 scrollBy()方法可以将浏览器窗口左上角定位到文档区域空间的某个位置，所观察的范围就是浏览器窗口的大小。

对于 body 对象的 innerHTML 和 innerText 的属性功能不同，如果一段字符串文本包含 HTML 标记，将该文本赋给 innerText 属性，则页面内容用 innerText 的文本替换，该文本不按照其包含的 THML 标记显示；如果同样的文本赋给 innerHTML 属性，则页面内容被 innerHTML 中的文本替换，该文本按照 HTML 规范解析。例如：document. body. innerText="< b > Hello "和 document. body. innerHTML="< b > Hello "，产生的结果不同。

从 HTML 文档结构看，body 对象是 document 对象的成员对象，对文档体内所有元素的访问应该通过 document. body 来访问。但是，为了书写上的方便，在 document 对象中，直接包含了有关文档体内的元素对象。例如：设置文档的背景色，理论上应该写为 document. body. bgColor="",但是，也可以直接写为 document. bgColor="",两种写法结果是一样的。实际上，对文档体中大多数元素的访问几乎都可以直接使用 document，而不是 document. body 来访问。

5.7.4 图像对象 Image

在 HTML 文档中，对应每个< img >标记，浏览器在内存中都创建一个对应的 Image 对象。与 document 对象和 body 对象不同，图像对象可能有多个，因此，可能给定一个唯一的对象名，因此，将图像对象定义为一个类，名称为 Image。每一个 img 元素都对应一个 Image 对象，这些 Image 对象保存在 document 对象的 images 数组成员中。

在 HTML 中，img 元素的属性很多，这也就决定了 Image 对象的属性，常用 Image 对象的属性见表 5-23。

表 5-23 Image 对象属性

属　　性	说　　明
id	存储 Image 对象 id 值
name	存储 Image 对象 name 属性值
src	存储 Image 对象对应的图片文件 URL
height, width	存储图片的高度和宽度
border	存储图片周围边框属性值
vspace	图片在垂直方向上与上面或下面文字之间的距离
className	存储图片对象的 class 属性值

在网页中使用图片，除了增加页面的视觉效果，还可以通过 JavaScript 程序动态地选择载入的图片，或者实现一些动画效果，从而增加页面的动感。

【例 5-11】 有两幅小狗左右张望的图片，利用 Image 对象，实现简单的动画效果。

分析：只要利用 window 对象的 setInterval()函数，两幅图像轮流显示即可。

代码清单：imgloop.htm

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
var imgnum = 0;
var imglist = new Array(2);
for (i = 0;i < 2;i++)
{
  imglist[i] = new Image();
  imglist[i].src = "dog" + (i + 1) + ".jpg"
}
////////////////////////////////////////////////////////////////////////////////
//对 dogimg 元素,设置对象的 src 属性,改变网页中显示的图片
function imgloop()
{
  imgnum = (imgnum + 1) % 2;
  dogimg.src = imglist[imgnum].src;
  form1.msg.value = imgnum;
}
var clockid = self.setInterval("imgloop()",1000);
</script>
</head>
<body>
<img id="dogimg" src="dog1.jpg">
<form name="form1">
  <input type="text" name="msg" size="15" />
  <input type="button" value="Stop" onclick="window.clearInterval(clockid)">
</form>
</body>
</html>
```

在创建 Image 对象时,使用 new Image(),其中的 Image 首字母是大写,这点特别要注意。在 JavaScript 中,内置对象一般采用首字母小写的小驼峰命名法,而对象类则采用首字母大写的大驼峰命名法。例如数组 Array 对象、日期 Date 对象等。

5.7.5 Link 对象与 Anchor 对象

在 HTML 文档中,有两种类型的链接,一种是在< head >..</head >之间的< link >元素,还有就是超链接元素< a >。对于< a >元素,一种用于定义一个超链接,即设置 href 属性,还有一种形式就是定义文档中的锚点(书签),即只有 name 属性,没有 href 属性。

根据上述情况,文档对象 document 有两个成员对象数组,即 anchors[]和 links[],anchors[]保存锚点(书签)对象,links[]保存 Link 对象和超链接对象。不论是 Link 对象还是 Anchor 对象,对象的属性总是与< link >元素和< a >元素的属性相对应。

【例 5-12】 设置和更改< a >元素的相关属性。

分析：对于文档中的超链接,使用 HTML DOM 对象,可以在已经显示的网页中动态地更改超链接的目标文件。

代码清单：links.htm

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<form>
    <input type="button" value="Google"
           onclick="document.links[0].href='http://www.google.com'">
    <input type="button" value="百度"
           onclick="document.links[0].href='http://www.baidu.com'">
</form>
<a href="javascript:alert('先单击按钮,选择一个搜索引擎,然后再单击搜索')">搜索</a>
<script>
document.write(document.links.length);
document.write(document.anchors.length);
</script>
</body>
</html>
```

在浏览器中打开上述页面，超链接<a>的href属性的初值为一段JavasSript代码，当用户单击超链接时，显示一个警示对话框。当用户单击了某个搜索引擎按钮后，将修改document.links[0].href的值，即a元素对应的内存对象的href属性，这样超链接的href就被设置成一个URL。然后，再单击超链接时，将打开某个搜索引擎网页。

5.7.6 表格对象 Table

在HTML中，表格是最重要的数据组织和页面布局工具，通过表格DOM对象操作，可以在客户端对表格进行插入行、删除行、移动行、修改单元格内容以及行和单元格的显示及隐藏等各种操作。特别是在数据库操作中，通过表格操作，可以减少客户端和数据库服务器的操作次数，减少服务器负载，优化服务器性能。

1. 表格对象

在网页文件中，可以包含多个表格。每一个<table>标记，浏览器都会在内存中创建一个表格对象，对象类的名称为Table。表格对象Table属性见表5-24。

表5-24 Table对象常用属性及方法

属　　性	
className	存储表格的class属性
width	存储表格的宽度，高度一般由表格行决定
border	设置或返回表格边框的宽度(以像素为单位)
cellSpacing	单元格之间的距离(以像素为单位)
cellPadding	单元格内容到单元格边框的距离(以像素为单位)
rows[]	表格包含的所有行对象(TableRow对象)构成的数组
cells[]	表格中所有单元格对象构成的数组。要定位一个i行，j列的单元格，可写为rows[i-1].cells[j-1]

续表

属　　性	
tBodies[]	表格中所有 tbody 对象构成的数组
方　　法	
createCaption()	用于在表格中获取或创建< caption >元素
deleteCaption()	从表格删除 caption 元素以及其内容
createTHead()	在表格中创建一个空的 tHead 元素
deleteTHead()	从表格删除 tHead 元素及其内容
createTFoot()	在表格中创建一个空的 tFoot 元素
deleteTFoot()	从表格删除 tFoot 元素及其内容
insertRow(index)	在表格 index 行之前插入一个新行,并返回新插入的行对象 TableRow。若 index 等于表中的行数,则新行将被附加到表的末尾。如果表是空的,则新行将被插入到一个新的< tbody >段,该段自身会被插入表中
deleteRow(index)	参数 index 指定要删除的行在表中的位置。行的编码顺序为行在文档源代码的出现顺序。< thead >和< tfoot >中的行与表中其他行一起编码

对于表格中的相关属性值,可以通过 window. alert(表格 id. 属性)来检查,例如:有一个表格的 id 为 Table1,则 window. alert(Table1. width)将显示表格的宽度值。同时,也可以看到 window. alert(Table1. height)显示“undefined”,表明表格对象不包含 height 属性。

2. 表格行对象

一个表格是由若干行(< tr >)构成的,每一行对应一个表格行对象,表格行对象类的名字为 TableRow。在 HTML 文档中,每一个< tr >元素,对应一个 TableRow 对象,其属性和方法见表 5-25。

表 5-25　表格行对象类 TableRow 常用属性及方法

属　　性	
rowIndex	存储表格行对象在集合中的位置(row index)
innerHTML	存储表格行开始标签和结束标签之间的 HTML 代码
cells[]	存储表格行包含的单元格对象数组
align	存储表格行中数据的水平排列设置
vAlign	存储表格行中数据的垂直排列方式设置
方　　法	
insertCell(index)	在单元格 index 之前插入一个新的单元格,并返回 TableCell 对象。若 index 等于行中的单元格数,则新单元格将被附加到行的末尾
deleteCell(index)	参数 index 指定了要删除的单元格在表中的位置

需要说明的是,虽然< tr >标记可以设置 height 属性,但表格行对象并不包含 height 属性,当然,也不包含 width 属性,执行 alert(Table1. rows[0]. height),显示“undefined”。

3. 单元格对象

单元格对象类的名称为 TableCell,一个 HTML 表格单元格对应一个 TableCell 对象。在一个 HTML 文档中,每一个< td >元素对应一个 TableCell 对象。TableCell 对象包含一组可读写的属性,但未定义方法,其常见属性见表 5-26。

表 5-26　单元格对象 TableCell 常用属性

属　　性	说　　明
id	存储单元格 id
width,height	存储单元格的宽度和高度值,如果单元格未设置 width 属性值或 height 属性值,则返回空
cellIndex	返回单元格在某行的单元格集合中的位置
innerHTML	存储单元格开始标签和结束标签之间的 HTML 代码
rowSpan,colSpan	存储单元格横跨的行数和列数
className	存储单元格的 class 属性
align	存储单元格内部数据的水平排列方式
vAlign	存储单元格内数据的垂直排列方式

在 HTML DOM 中,单元格对象包含了高度和宽度属性,一般情况下,我们会在< tr >标记中设置行的高度,但表格行对象和表格对象都不包含 height 属性。

【例 5-13】 设计一个页面,实现表格行顺序的上下调整。

分析:在 Web 开发中,对于列表项,有时候我们需要调整它们的显示顺序。显示顺序的调整最简单的方法是设计一个顺序编号,在从数据库读出时,按顺序编号排序。另一种办法就是,允许用户在客户端调整。

设我们设计的一个问卷项目编辑及项目顺序调整页面如图 5-12 所示。

【课程问卷管理】A(50 分)二级指标管理　　添加二级指标项目

序号	二 级 指 标	操　作		显示顺序	
1	老师讲课条一清楚,深入浅出(10)	修改	删除	上移	下移
2	老师为我们耐心答疑,并且时间充分(10)	修改	删除	上移	下移
3	我认为老师对我们和蔼可亲,平易近人(10)	修改	删除	上移	下移
4	中心网站——学习园地上的内容对我们学习有帮助(10)	修改	删除	上移	下移
5	教师教学认真,对我们要求严格(10)	修改	删除	上移	下移
单击页面右上角的"添加二级指标项目",可以为一级指标添加二级指标项目					

图 5-12　问卷调查项目编辑界面

实现上述功能代码清单如下:surveymodal-edit2.htm

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script type="text/javascript">
////////////////////////////////////////////////////////////////////////////////
//编辑一项新的项目
function itemnew()
{
    var tempstr="<p align=left>二级指标:<input type='text' name='itemname' size='35'>";
    tempstr +="分值:<input type='text' name='score' value='0' size='2'>";
    tempstr +="<input type='button' name='btn1' value='添加' onclick='itemadd()'>";
    tempstr +="<input type='button' name='btn2' value='取消' onclick='itemcancel()'></p>";
```

```
    editarea.innerHTML = tempstr;
}
////////////////////////////////////////////////////////////////////////////
//上移项目,交换第 2 列单元格对象,itemcode 为第 2 列单元格用户自定义属性
function moveup(itemnum)
{
    var rownum = parseInt(itemnum);
    if (rownum == 1) return 0;
    var temp = table1.rows[rownum - 1].cells[1].innerText;
    table1.rows[rownum - 1].cells[1].innerText = table1.rows[rownum].cells[1].innerText;
    table1.rows[rownum].cells[1].innerText = temp;
    var itemcode = table1.rows[rownum - 1].cells[1].itemcode;
    table1.rows[rownum - 1].cells[1].itemcode = table1.rows[rownum].cells[1].itemcode;
table1.rows[rownum].cells[1].itemcode = itemcode;
}
////////////////////////////////////////////////////////////////////////////
//下移项目
function movedown(itemnum)
{
    var rownum = parseInt(itemnum);
    var temp = table1.rows[rownum + 1].cells[1].innerText;
    table1.rows[rownum + 1].cells[1].innerText = table1.rows[rownum].cells[1].innerText;
    table1.rows[rownum].cells[1].innerText = temp;
    var itemcode = table1.rows[rownum + 1].cells[1].itemcode;
    table1.rows[rownum + 1].cells[1].itemcode = table1.rows[rownum].cells[1].itemcode;
    table1.rows[rownum].cells[1].itemcode = itemcode;
}
</script>
</head>
<body>
<table id = "table1" border = "1px" style = "border - collapse:collapse;empty - cells:show;">
<tr height = "35px">
    <td width = "10 %">序 号</td>
    <td width = "70 %">二级指标<a href = "#" onclick = "itemnew()">添加项目</a></td>
    <td colspan = "2">显示顺序</td>
</tr>
<tr height = "30px">
    <td>1 - 1 </td>
    <td itemcode = "item11">老师讲课条理清楚,深入浅出(10 分)</td>
    <td><a href = "#" onclick = "moveup('1')">上移</a></td>
    <td><a href = "#" onclick = "movedown('1')">下移</a></td>
</tr>
<tr height = "30px">
    <td>1 - 2 </td>
    <td itemcode = "item12">耐心答疑,时间充分(10 分)</td>
    <td><a href = "#" onclick = "moveup('2')">上移</a></td>
    <td><a href = "#" onclick = "movedown('2')">下移</a></td>
</tr>
<tr height = "60px">
    <form name = "form1" method = "post" action = "surveymodal - edit2save.jsp" target = "_self">
    <input type = "hidden" name = "head2list" value = "<% = head2list %>">
```

```
        <td id="editarea" colspan="4">
         单击页面右上角的"添加二级指标项目",可以为一级指标添加二级指标项目。
        </td>
        </form>
    </tr>
    </table>
    </body>
    </html>
```

上述代码演示了客户端强大的程序功能,这可以减少和 Web 服务器的连接次数,减少服务器的负载,提高整个系统的性能。上述示例程序,有两处核心代码:

(1) 在表格第 2 列中,我们在< td >标记中设置了一个用户自定义属性 itemcode,这是一种全新的用法,因为在 HTML 规范中,< td >标记中无此属性。对于有的标记,我们不仅需要标记的内容(即 innerText 和 innerHTML),有时候还需要更多的信息,这些信息希望不显示在单元格中,故为< td >添加了一个用户自定义属性 itemcode。

从 HTML DOM 对象的思想出发,对于 HTML 中的标记,除了规范给定的属性外,用户还可以根据需要设定标记属性,JavaScript 都将其转化为 DOM 对象的属性,这就大大提高了标记和 HTML DOM 的灵活性。这种用法,还常常用于复选框的分类等。

(2) 对于添加项目函数 itemnew(),示例代码没有给出其中涉及的所有函数,但它演示了 Web 开发中数据维护的三个主要问题:添加、修改和删除,图 5-12 给出了一个很好的功能组织界面,界面友好始终是 Web 系统设计和开发的重要内容。

(3) 为了突出重点,上述 HTML 代码只列出了核心代码,对于表格样式没有给出。其中,最重要的是表格和单元格线条问题,要显示表格中单元格的线条,必须设置表格属性 border="1px",该属性不能设置线型和颜色。如果要设置表格线的颜色和线型,则需要设置表格的 CSS 属性 border,例如:style="border:1px solid #0000ff;",两个属性不能合并。

5.7.7 表单对象 Form

在 Web 页中,表单(form)是人机交互的主要手段。在一个网页中,可以包含多个表单,每一个< form >元素都在内存中创建一个表单对象,对象类名为 Form。Form 对象封装了网页中的 form 元素,Form 对象类属性及方法见表 5-27。

表 5-27 Form 对象类常用属性及方法

一 般 属 性	
name	存储表单名称,即 form 元素的 name 属性值
method	存储数据发送到服务器的 HTTP 方法
enctype	存储表单用来编码内容的 MIME 类型
action	存储表单的 action 属性值
target	存储表单提交后 action 输出结果的输出窗口,默认值为当前页面窗口
length	存储表单中的元素数目
elements[]	数组成员,存储表单中所有元素对象

续表

事 件 属 性	
onsubmit	当用户单击表单中的 Submit 按钮提交一个表单时执行该方法,执行完后提交表单。如果不设置该属性,则直接提交表单
onreset	在重置表单元素之前调用
方　　法	
reset()	把表单的所有输入元素重置为它们的默认值
submit()	提交表单,调用 form 的 onsubmit()方法

在表单输入中,在提交表单以前,通常要进行数据有效性验证。因此,在实际编码时,如果使用 Submit 按钮提交数据,则需要在 form 中添加 onsubmit 属性,在处理函数中完成数据有效性验证工作。

很多情况下,可能使用普通 button、超链接或任何一个可接受单击的元素来提交表单,此时,需要定义 onclick 处理函数,在处理函数中首先进行数据输入有效性验证,最后执行 document. form-name. submit()提交表单。表单提交后,form 的 action 属性设置的服务端页面被调用执行。

在< form >中通常包含了一系列输入元素,它们也被创建为相应的 DOM 对象,并被保存到 Form 对象的数组成员 elements[]中。对于输入对象的属性和方法,与输入标记的属性对应,在此不再详细介绍。

1. 访问 Form 对象

从 HTML 文档结构看,表单是文档对象 document 的成员。在一个 HTML 文档中,可以定义多个表单,因此在 JavaScript 中,访问 Form 对象有多种实现方法。

方法一:通过 form 名称访问

在< form >标记中,包含一个 name 属性,它对应了 Form 对象的对象名,JavaScript 程序可以通过表单名来访问表单对象,即:

```
obj = document. form - name
```

当然也可以利用 document. getElementsByName("form-name")来返回 Form 对象。

方法二:通过 document 对象 forms[]属性访问

文档对象 document 包含 forms[]数组成员,存储了文档中定义的所有表单元素。因此,可以通过 document 对象的 forms 对象数组来访问 Form 对象,有两种用法:

document. forms["form-name"]或 document. forms[num],num 为 0,1,...整数,都返回 Form 对象。需要说明的是,在 JavaScript 中要对 form 引用的条件是:必须先在页面中用< form >标记创建表单,并将定义表单部分放在引用之前。

2. 访问 form 中的元素

一个表单是由若干的表单输入元素组成的,这些输入元素包括:文本框(text)、单选按钮(radio)、复选框(checkbox)、按钮(button)等。在 HTML 5 中,又新增了输入类型:日期(date)、时间(time)、日期时间(datetime)、月份(month)、周(week)、数字(number)、范围(range)、图片(image)、邮件地址(Email)、网址(url)等。HTML 5 丰富的输入类型,使得用户输入的有效性验证编码变得更加简单。

对于每一个输入元素，都在内存中创建相应的内存对象。根据 HTML 文档层次结构，访问表单元素 DOM 对象的方法有以下几种形式：

方法一：直接通过输入元素名

Obj＝document. form-name. element-name

方法二：通过 Form 对象的 elements[]数组

每一个 Form 对象包含一个 elements[]数组成员，存储了表单中所有的输入元素的 DOM 对象，因此，可以通过访问 elements[]数组来得到一个输入对象。一般形式是：

```
Obj = document. form - name. elements["element - name"]
```

两种方法都返回表单输入元素对应的 DOM 对象，不同的 form 元素，对应的内存对象的属性和方法也不相同，但每一个内存对象包含的属性和方法均与其对应的 HTML 标记对应。例如，对于 text 内存对象，对应一个单行文本框输入元素。在 HTML 中，标记单行文本框 text 输入的一般形式是：

```
< input type = "text" name = "input - name" value = "default - value">
```

因此，对应于每一个 text 对象，其包含的属性包括：name 属性、value 属性等。对对象属性的操作，即通过对象的点操作符完成，即：对象名.[属性|方法]。

对于每一个输入对象，通常还包含一组通用方法，如 blur()方法，将当前焦点移到后台；select()方法，选择 text 框内的文本。包含的事件有：onFocus：当 text 获得焦点时，产生该事件。onBlur：从元素失去焦点时，产生该事件。onselect：当文字被选中，产生该事件。onchange：当元素值改变时，产生该事件。

【例 5-14】 设计一个用户信息输入页面，完成用户个人信息的填写，当用户提交表单时，显示用户的输入信息。

分析：在上网过程中，数据输入大都是通过表单来完成的，如何提取表单中的用户输入数据，包括：文本框、复选框、单选钮、列表框、多行文本框等输入数据，以及对输入进行控制和有效性验证是 Web 编程中的共性和基础性问题。

设有一个用户信息输入界面，如图 5-13 所示。

图 5-13　表单输入界面

在上述表单中,包括了常用的元素输入类型。除了要获取输入元素的数值外,经常还需要对用户输入进行控制,例如:检查文本框输入数据的合理性,设置文本框的只读属性,限制文本输入的字符个数,等等。

对应上述输入界面,代码清单如下:

代码清单:person-add.htm

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
body{margin-top:10px;background-color:#FFF8EB;font-size:30px;}
td{font-size:13px}
</style>
<script type="text/javascript">

////////////////////////////////////////////////////////////////////////////////
//返回单选按钮的值
function getsexval()
{
  var sexValue="";
  for (i=0;i<form1.sex.length;i++)
  {
     if (form1.sex[i].checked)
     {
        sexValue=form1.sex[i].value;
        break;
     }
   }
  return sexValue;
}
////////////////////////////////////////////////////////////////////////////////
//获取复选框选择,多个复选框元素可以设置相同的 name 属性
function getcheckboxval()
{
    var likestr="";
    var length=document.form1.mylike.length;
    for(i=0;i<length;i++)
    {
        if(document.form1.mylike[i].checked)
          likestr += document.form1.mylike[i].value+";";
    }
    return likestr.substring(0,likestr.length-1);
}
////////////////////////////////////////////////////////////////////////////////
//获取下拉列表输入,取<optin value=></option>中 value 的值,或标记的文本
function getlistboxval(obj)
{
   return obj.options[obj.selectedIndex].text;
}
```

```
////////////////////////////////////////////////////////////////////////////////
//限制多行文本域输入的字符个数,避免超过数据库字段长度
function CountUpdate(strobj)
{
  var strlength = strobj.value.length;
  var maxnum = parseInt(document.form1.total.value);
  if (strlength > maxnum)
  {
    alert("个人简介最多不超过" + maxnum + "个字符!");
    strobj.value = strobj.value.substring(0,strlength - 1);
    strlength -= 1;
  }
  document.form1.used.value = strlength;
  document.form1.remain.value = maxnum - strlength;
}
////////////////////////////////////////////////////////////////////////////////
//验证身份证输入位数是否正确
function checkuserid(str)
{
   var Expression = /\d{15}|\d{17}[a - zA - Z]/;
   var objExp = new RegExp(Expression);
   if (objExp.test(str) == true) return true;
   else
   {
      alert("身份证位数应该为 15 位或 18 位!");
      return false;
   }
}
////////////////////////////////////////////////////////////////////////////////
//验证邮箱地址是否正确
function checkemail(str)
{
   if (str == "")
   {
     alert("用户邮箱不能为空!");
     return false;
   }
   //在 JavaScript 中,正则表达式只能使用"/"开头和结束,不能使用双引号
   var Expression = /\w + ([ -+ .']\w + ) * @\w + ([ - .]\w + ) * \.\w + ([ - .]\w + ) * /;
   var objExp = new RegExp(Expression);
   if (objExp.test(str) == true) return true;
   else
   {
      alert("用户邮箱格式不正确!");
      return false;
   }
}
////////////////////////////////////////////////////////////////////////////////
//提交表单前对输入数据进行有效性验证,最后提交表单
//在< form1 >中,未设置 action 属性,程序最终是通过 form99 提交到服务器端的
function form1submit()
```

```
{
  //用户账户不能包含汉字字符
  if (escape(document.form1.useraccount.value).indexOf("%u")!= -1)
  {
     alert("用户账户不能包含汉字,请重新输入");
     form1.useraccount.focus();
     return false;
  }
  //密码由 6-20 位的字母、数字、下画线、句点组成
  var Expression=/^[A-Za-z0-9]{1}([A-Za-z0-9]|[._@]){5,19}$/;
  var objExp=new RegExp(Expression);
  if (objExp.test(form1.password.value)==false)
  {
     alert("密码由 6-20 位的字母、数字、点、下画线、@组成且首字符为字母");
     form1.password.focus();
     return false;
  }
  //检查姓名不能为空
  if (form1.username.value=="")
  {
     alert("姓名不能为空!");
     form1.username.focus();
     return false;
  }
  //检查身份证号码的有效性
  if (!checkuserid(document.form1.userid.value))
  {
     form1.userid.focus();
     return false;
  }
  //检查 Email 的有效性
  if (!checkemail(document.form1.email.value))
  {
     form1.email.focus();
     return false;
  }
  //提取各输入元素的值
  var str="个人基本信息:\n";
  str += "姓名:" + document.form1.username.value+"\n";
  str += "性别:" + getsexval() + "\n";
  str += "教育状况:" + getlistboxval(form1.grade) + "\n";
  str += "个人兴趣:" + getcheckboxval() + "\n";
  str += "个人简介:" + document.form1.mybrief.value+"\n";
  //将得到的数据合并成一个串,赋给 form99 的 hidden 元素,提交 form99
  form99.totalinfo.value = str;
  form99.submit();
}
</script>
</head>
<body>
<form name="form1">
```

```
<table border = "0" style = "border:1px solid red;padding - left:10px">
<tr height = "35">
    <td colspan = "2"><img src = "square.gif" valign = "absmiddle"> 用户个人资料</td>
</tr>
<tr height = "1">
    <td colspan = "2"><img src = "line2.gif"></td>
</tr>
<tr height = "30">
   <td align = "right">用户账户: </td>
   <td>
    <input type = "text" name = "useraccount" value = "haoxw365" disabled>(用户账户不能自行
修改)
   </td>
</tr>
<tr height = "30">
   <td width = "100" align = "right">姓名: </td>
   <td>
       <input type = "text" name = "username" value = "">性别
       <input type = "radio" name = "sex" value = "male" checked>男
       <input type = "radio" name = "sex" value = "female"> 女
   </td>
</tr>
<tr>
    <td align = "right">教育状况: </td>
    <td align = "left">
       <select name = "grade">
          <option value = "博士">博士</option>
          <option value = "硕士">硕士</option>
          <option value = "本科" selected>本科</option>
       </select>
    </td>
</tr>
<tr height = "30">
   <td align = "right">身份证号码: </td>
   <td><input type = "text" name = "userid" maxlength = "18"></td>
</tr>
<tr height = "30">
   <td align = "right">个人邮箱: </td>
   <td><input type = "text" name = "email" size = "45"></td>
</tr>
<tr>
    <td align = "right">个人兴趣: </td>
    <td> 
        <input type = "checkbox" name = "mylike" value = "体育">体育
        <input type = "checkbox" name = "mylike" value = "音乐">音乐
        <input type = "checkbox" name = "mylike" value = "旅游">旅游
    </td>
</tr>
<tr height = "30">
   <td align = "right">自我介绍: </td>
   <td>
```

```
        < textarea name = "mybrief" cols = "45" rows = "6" onKeyDown = "CountUpdate(this);"
onKeyUp = "CountUpdate(this);"></textarea >
    </td >
</tr >
< tr >
    < td colspan = "2" align = "center">最多允许
        < input type = "text" name = "total" value = "100" disabled >个字节 已用字节:
        < input type = "text" name = "used" value = "0" disabled >剩余字节:
        < input type = "text" name = "remain" value = "100" disabled >
    </td >
</tr >
< tr height = "30">
    < td colspan = "2" align = "center">
        < input type = "button" name = "myok" value = "确定" onclick = "form1submit();">
        < input type = "reset" name = "mycan" value = "取消">
    </td >
</tr >
</table >
< input type = "hidden" name = "password" value = "111111">
</form >
</table >
< form name = "form99" method = "post" action = "personaddsave.jsp">
< input type = "hidden" name = "totalinfo" value = "">
</form >
</body >
</html >
```

对于上述代码,说明如下:

(1) 在表格中,设置< table >标记属性 border="1",表格、单元格都含有表格线。若< table>标记属性 border="0",则单元格无边框。设置< table >的内联样式 style="border:1px solid red",只用于表格的四个边框,不用于单元格。

(2) 表单提交按钮设计为 Button 类型,而不是 Submit 类型,主要是增加可读性,在 form1submit()中进行数据的有效性验证。在实际开发中,一个网页通常不止一个表单,每个表单都可以设置 action 属性,提交表单,服务端将执行相应的服务器页。有时候需要将多个表单数据赋给一个表单,一起提交,这就是上述代码 form99 的用途。

(3) 对于 JavaScript 函数,参数基本上有两类,一类是字符串,另一类是对象。例如 checkemail(),如果参数是对象,则调用时可使用 checkemail(form1. email),如果是在标记的事件属性中调用,可直接写 checkemail(this),在函数内部再取对象的 value 值。如果参数是字符串,实际参数就写为 form1. email . value,this. value,即当前对象的 value 值。

(4) 表单中使用了 hidden 输入,主要用于客户端和服务端之间的数据交换,如果是服务器页(如 JSP),很容易把服务端的数据通过 hidden 传递到客户端。例如:< input type= hidden name="nowdate" value="<%=new date()%>",这样客户端就得到了服务器上的时钟。

(5) 数据有效性验证是一项重要的工作,可以避免在数据库操作时,出现数据插入、更新失败等操作,比如用户实际的数据比数据库字段定义的长度更长,在服务端则会引起数据

插入操作失败。

5.7.8 事件对象 event

在网页浏览中，会有各种各样的鼠标和键盘操作，称为事件(event)。在 JavaScript 中定义了一个事件对象 event，封装了相关的键盘和鼠标操作，包括键盘按键的状态、鼠标的位置、鼠标按钮的状态等信息。

1. event 对象常用属性

事件对象 event 常用属性见表 5-28。

表 5-28 事件对象 event 常用属性

属 性	说 明
clientX,clientY	存储鼠标事件触发时，鼠标指针的 X、Y 坐标
screenX,screenY	存储鼠标事件触发时，相对于屏幕的横坐标和列坐标
button	存储鼠标事件触发时被单击的鼠标按键，0、1、2 分别代表鼠标左键、鼠标中键和鼠标右键
ctrlKey	存储事件发生时，Ctrl 键的状态，按下为 1，否则为 0
altKey	存储事件发生时，Alt 键的状态，按下为 1，否则为 0
shiftKey	存储事件发生时，Shift 键的状态，按下为 1，否则为 0
target	存储事件目标节点对象(触发该事件的节点)，如生成事件的元素、文档或窗口
keyCode	对于 keypress 事件，存储被敲击的键的 Unicode 字符码。对于 keydown 和 keyup 事件，存储被敲击的键的虚拟键盘码
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

2. 可能的事件

在浏览器中可能的事件主要有：onclick(鼠标单击)、ondblclick(鼠标双击)、onmousedown(按下鼠标按钮)、onmousemove(鼠标移动)、onmouseout(鼠标从某元素移开)、onmouseover(鼠标移到某元素之上)、onmouseup(松开鼠标按键)、onkeydown(按下键盘按键)、onkeypress(按下并松开键盘按键)、onkeyup(松开键盘按键)、onfocus(元素获得焦点)、onblur(元素失去焦点)、onchange(改变域的内容)、onselect(选中文本)、onload(网页或一幅图像完成加载)、onunload(退出页面)、onabort(图像加载中断)、onerror(加载文档或图像时出错)、onresize(调整窗口或框架的大小)、onreset(单击重置按钮)、onsubmit(单击确认按钮)等。

【例 5-15】 编写输入元素事件处理函数，当按回车键时，将输入焦点自动移动到下一个输入元素。

说明：在表单输入中，默认情况下，按回车键，输入焦点不会发生变化。这不符合我们的交互习惯，我们的一般习惯是：按回车键后，输入焦点将自动移向下一个表单元素。

利用事件对象 event，当输入字符为回车键时，转换输入焦点，代码如下：

```
<script type="text/javascript">
function myenter(nextobj)
{
  if (event.keyCode == 13)
    nextobj.focus();
```

```
}
</script>
```


然后在每一个 Input 元素中,可以修改 onkeypress 事件属性为 onkeypress ="myenter(下一个输入元素)"。例如:

```
<form name = "form1">
<input type = "text" name = "useraccount" onkeypress = "myenter(form1.password)" />
<input type = "text" name = "password" onkeypress = "myenter(form1.username)" />
</form>
```

则用户输入完用户账户 useraccount 后,按回车键,输入焦点将转到 name="password" 的输入框,输入完 password 后,转到 username 输入框。

5.7.9 应用举例

在 Web 系统的开发中,表格和表单是使用最多的页面元素,利用 JavaScript 可以动态地在表格中插入、删除表格行,同时也可以隐藏表格行和单元格。实现一种类似数据库中记录插入和删除操作的用户界面,最后一次性提交到 Web 服务器。

在我们进行 GSL 系统的研发中,设计了考试模型编辑界面,对于每一类题,列出题库中的题目,选择需要的题目,并给定分值,最后提交,界面设计如图 5-14 所示。

图 5-14 考试模型编辑页面

分析:采用< iframe >来显示下部的列表,正常运行时,列表来自数据库查询所得数据。在列表项右侧,单击“添加”按钮,则相应的项目添加到上部的已选题目列表,对已选题目列表中的题目,如果要放弃选择,单击右侧的“取消选定”即可。最后单击“确定”按钮,将所选题目列表发送到服务器,更新数据库中课程考试模型数据。

界面设计两个 HTML 文件,一个为主体页面对应的 HTML 文件,文件名为 selelist.htm,另一个为< iframe >中显示的文件,文件名为 datalist.htm。

(1) 已选题目页面 selelist.htm 代码清单:

```
<html>
```

```
<head>
<meta charset = "utf - 8">
<script>
////////////////////////////////////////////////////////////////////////////////
//动态在表格中添加一行
//子窗口调用的函数,必须单独写在一个独立的脚本定义段内,否则出错
function addonerow(str)
{
   var rowsnum = document.all.table1.rows.length;
   var newRow,newCell;
   //在最后一行的前面插入一行,每行 4 个单元格,并设置行的 id 属性
   var newRow = table1.insertRow(rowsnum - 1);
   newRow.id = "row" + rowsnum;
   for(var j = 0;j < 4;j++)
   {
       newCell = newRow.insertCell(j);
   }

   //为新添加的行的各个单元格赋值
   var s = new Array(2);
   var ch1 = "\10";
   s = str.split(ch1);
   newRow.cells[0].innerText = s[0];
   newRow.cells[0].width = "15 % ";
   newRow.cells[0].align = "left";

   newRow.cells[1].innerText = s[1];
   newRow.cells[1].width = "50 % ";
   newRow.cells[1].align = "left";

   newRow.cells[2].innerHTML = "分数: < input type = 'text' name = 'score' size = '3' value = '0'
onblur = \"checkscore(this)\">";
   newRow.cells[2].width = "10 % ";
   newRow.cells[2].align = "center";

   newRow.cells[3].innerHTML = "< input type = 'button' name = 'btn' value = '取消选定' onclick =
\"cancelselect('row" + rowsnum + "')\">";
   newRow.cells[3].align = "center";
   //为单元格设置样式
   newRow.cells[0].style.cssText = "font - size:15px;color:blue;";
   newRow.cells[1].style.cssText = "font - size:15px;color:blue;";
   newRow.cells[2].style.cssText = "font - size:15px;color:blue;";
   //移动滚动条到窗口底部,也可用 window.scrollBy()
window.scrollTo(0,document.body.scrollHeight);
}
</script>

<script>
////////////////////////////////////////////////////////////////////////////////
//检查输入的分数是否合适
function checkscore(obj)
```

```
{
    var scorestr = obj.value;
    var Expression = /^\d{1,3}$/;
    var objExp = new RegExp(Expression);
    if (objExp.test(scorestr) == false)
    {
        alert("数字为不大于 100 的整数!");
        obj.focus();
        return false;
    }
    return true;
}
////////////////////////////////////////////////////////////////////////////////
//隐藏一个 id 所标记的区域,如 div、表格行、单元格等
function cancelselect(idstr)
{
  if (document.all[idstr].style.display == "none")
      document.all[idstr].style.display = "block";
  else
      document.all[idstr].style.display = "none";
}

////////////////////////////////////////////////////////////////////////////////
//将用户选择的题目,形成 itemlist 列表传回服务器保存到数据库
//
function form1submit()
{
    var rowsnum = document.all.table1.rows.length;
    //只有三行,第一行是提示行,第二行为标题行,最后一行为"确定"按钮,则表明无数据
    //rownums 包括隐藏的行
    if (rowsnum == 3)
       return false;
    //第一行、第二行、最后一行不是题目数据,隐藏的行不提交
    //将表格中的内容形成一个考试题串,题目内部分用"\10"分隔,题目之间用"\20"分开
    var tempstr = "",s1,s2,s3;
    var i = 0,totalscore = 0;
    for (i = 2;i < rowsnum - 1;i++)
    {
        if (table1.rows[i].style.display!= "none")
        {
            s1 = table1.rows[i].cells[0].innerText;
            s2 = table1.rows[i].cells[1].innerText;
            //当记录只有一行时(包含隐藏行),score 未形成数组
            if (rowsnum == 4)
                s3 = form1.score.value;
            else
                s3 = form1.score[i - 2].value;
            tempstr += s1 + "\10" + s2 + "\10" + s3 + "\20";
            totalscore += parseInt(s3,10);
         }
    }
```

```
    //原先有数据,但取消了所有的行,即指定的题目列表变为空
    if (tempstr!= "" && totalscore!= 100)
    {
       alert("已选题目总分为: " + totalscore + ",总分应为 100 分")
       return false;
    }
    form1.itemlistnew.value = tempstr;
    form1.submit();
}
</script>
</head>
<body style = "margin-top:0px;">
<form name = "form1" action = "tableformmainsave.jsp">
<input type = "hidden" name = "itemlistnew" value = "">
<table id = "table1" width = "100%">
<tr height = "35" id = "row0">
    <td colspan = "4">已选题目</td>
</tr>
<tr height = "35" id = "row1">
    <td width = "15%">题目代码</td>
    <td width = "50%">题目名称</td>
    <td width = "10%">分值比例</td>
    <td>操 作</td>
</tr>
<tr height = "30">
   <td colspan = "4">
       <input type = "button" name = "btn2" value = "确定" onclick = "form1submit()">
       <input type = "button" name = "btn2" value = "取消">
   </td>
</tr>
</table>
</form>
<table id = "table2" width = "100%">
<tr>
   <td>
     <iframe name = "datalistframe" src = "datalist.html" width = "100%" height = "330"
frameborder = "0" scrolling = "yes">
     </iframe>
   </td>
</tr>
</table>
</body>
</html>
```

(2) <iframe>内显示的文件 datalist.htm,代码清单如下:

```
<html>
<head>
<meta charset = "utf-8">
<script>
////////////////////////////////////////////////////////////////////////////////
```

```
//一个网页只要是打开的,它的 DOM 对象就可以访问.只要清楚窗口之间的关系,
//就可以实现不同窗口之间的互操作
//例如,本例中实现 iframe 内部页面直接调用该 iframe 所属父窗口自定义函数的方法
//不同的浏览器支持不同,IE 支持下列代码,在 Chrome 中不支持
function toaddonerow(itemnum)
{
    var ch1 = "\10"; //定义一个分隔符,分隔一行中不同列数据
    str = eval("item" + itemnum + "1").innerText + ch1;
    str += eval("item" + itemnum + "2").innerText;
    //调用父窗口函数,在已选列表中新添加的题目显示
    window.parent.addonerow(str);
}
</script>
</head>

<body>
<table class = "table_frame" width = "100 % " cellpadding = "0" cellspacing = "0">
<tr height = "35">
    <td width = "15 % ">题目代码</td>
    <td width = "75 % ">题目名称</td>
    <td>操 作</td>
</tr>
<tr height = "30">
    <td id = "item11"> 201107001 </td>
    <td id = "item12"> Line1...</td>
    <td><a href = " # " onclick = "toaddonerow('1');return false;">添加</a></td>
</tr>
<tr height = "30">
    <td id = "item21"> 201107002 </td>
    <td id = "item22"> Line2...</td>
    <td><a href = " # " onclick = "toaddonerow('2') ;return false;">添加</a></td>
</tr>
</table>
</body>
</html>
```

上述页面较好地使用了表单、表格操作,同时用到了< iframe >等技术,在 Web 开发中具有较好的参考价值。主要技术点有:

(1) 子窗口与父窗口网页之间的互操作,一个页面中调用另一个页面的函数的方法,以及为另一个页面中变量的赋值。不同浏览器对窗口互操作的实现不同,本例中的代码在 IE 中正常运行,在 Google Chrome 浏览器中,添加按钮对应的调用父窗口的函数不能正常运行。

(2) 对于< form >中输入元素 name 属性重名问题,在 JavaScript 中,允许多个输入域具有相同的 name 值。例如:本题中的分数,如果有两行或以上,则有多个 name＝"score"的输入文本框,对于多个 score,JavaScript 将创建一个数组对象来存储,数组名为 score。但是,在系统运行时,如果只有一行,则只能创建一个简单的 DOM 对象 score,而不是数组。这是在程序调试时最容易遇到的问题。

(3) 表格的动态处理,添加行、删除行和隐藏行都可以通过 Table 对象方法来实现。单元格样式设置是一个难点,各个列的样式类似,但必须分别设置。

5.8 网页异步通信 AJAX 技术

在 Web 系统中，前端和后端的交互是通过提交表单完成的。当提交表单后，客户端表单数据被发送到服务端，由服务端程序进行处理，并返回处理结果，在客户端显示。这个过程是在 form 元素中设定的，其 action 属性指定了接受客户端数据的服务端程序页面，target 属性则指定了服务端程序输出的显示窗口，这些输出被发送到客户端指定的窗口显示。

在两个页面的交互中，传统的服务端输出的目标窗口通常是当前窗口，即覆盖客户端页面，当然也可以指定其他的输出窗口。有时候，我们不能覆盖客户端的整个窗口，而需要仅仅更新客户端页面的局部，AJAX 技术就是为此目标而设计的，它广泛应用于许多需要实时刷新页面局部的应用中。

5.8.1 AJAX 的概念

在 Web 应用中，用户在网页上输入数据时，单击“提交”按钮，客户端浏览器则把这些信息发送到服务器端，服务器根据用户的操作发送一个新页面到客户端。例如，在一个登录页面中，当用户提交表单后，服务器将比较用户在表单中输入的数据与数据库中保存的登录信息是否一致。如果用户输入的数据不正确，服务器就把与原来相同的登录页面重发给用户，而这个页面与原来的页面相比可能只是多了“登录失败”的消息。用户每发出一个请求，整个网页就要被刷新一次，即页面的加载与用户的请求是同步的。

刷新整个页面除了带来较大的网络流量外，对于一些聊天类的网站，频繁的页面刷新必然会产生闪烁，影响用户的视觉体验。2005 年 2 月，Jesse James Garrett[①] 在一篇文章中提出了 AJAX 技术，即 Asynchronous JavaScript And XML(异步 JavaScript 和 XML)的缩写，AJAX 提供与服务器异步通信的能力，一个最简单的应用是无须刷新整个页面而在网页中更新一部分数据。

如果采用 AJAX 技术，当用户提交表单后，如果登录失败，将不再刷新整个网页，而是仅仅在页面上增加了“登录失败”的消息文本，即 AJAX 技术可以实现网页的局部刷新，而页面上的所有没有被刷新的信息都是提交表单前页面的内容。这无论是对于服务器的 CPU 开销还是对网络的传输开销，无疑都减轻了不少压力，也避免了页面闪烁现象的发生。

5.8.2 XMLHttpRequest 对象

从原理上讲，AJAX 技术不是一项新的技术，它和传统的 Web 不相同的是在浏览器端增加了一种新的响应层，而不是传统的“提交网页-返回网页”同步模式。它只是在一个网页上做操作，操作发送到服务端，服务端的返回只是更新网页的一个局部，就如传统的桌面程序一样。AJAX 技术主要涉及 JavaScript、DOM、XML 和 HTTP 等内容。

---

① Jesse James Garrett，美国用户体验咨询公司 Adaptive Path 联合创始人。出版《用户体验要素：以用户为中心的 Web 设计》(*The Elements of User Experience: User-Centered Design for the Web*)一书，提出从抽象到具体 5 个层级的概念。他在用户体验领域的贡献还包括“视觉词典(the Visual Vocabulary)”，一个为规范信息架构文档而建立的开放符号系统，该系统在全球众多企业中得到广泛应用。

1. XMLHttpRequest对象

1999年春,在IE 5.0中,增加了一个新的ActiveX控件,即XML HTTP请求对象XMLHttpRequest(XHR),这个对象主要在IE中使用,用于向服务端发出异步通信请求。随后,其他浏览器开始支持XHR对象,并成为W3C标准的一部分。XMLHttpRequest对象位于客户端浏览器中,是用来实现网页与Web服务器之间异步通信的对象,通过它可以在不进行整个网页刷新的情况下向服务器发出请求、接收响应等工作。

XMLHttpRequest对象提供了对HTTP协议的完全访问,包括做出POST和HEAD请求以及普通的GET请求的能力。XMLHttpRequest可以同步或异步返回Web服务器的响应,并且能以文本或者一个DOM文档形式返回内容。XMLHttpRequest对象是AJAX的Web应用程序架构的一项关键功能。AJAX技术工作机制如图5-15所示。

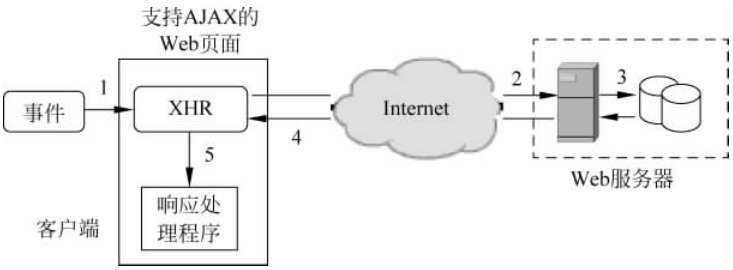

图5-15 AJAX技术工作机制

对于一个支持AJAX的Web面来说,与服务器进行异步数据通信的过程如下:

(1) 当要求进行与服务器异步通信的某一事件发生时,事件处理程序将调用XHR,设置该对象相关的属性参数,如指定服务端程序页面,设置服务器返回数据处理函数。

(2) 由XHR向服务器发出请求,请求通过Internet发送到Web服务器。当服务器收到请求后,运行指定的服务器端程序。

(3) 服务器端程序执行中如果包含数据库访问的命令,则连接数据库服务器完成数据库的相关操作,结果返回到Web服务器。

(4) Web服务器将服务器页面输出结果通过Internet返回到客户端,浏览器将这些响应信息交给Web页面的XHR对象,由相应的处理函数处理服务器返回的数据。

(5) 浏览器启动相应的处理程序,通过文档的DOM模型完成页面的更新工作,例如:将返回结果更新DOM元素。

采用AJAX技术,其核心是XML HTTP请求对象XMLHttpRequest。XMLHttpRequest是一种浏览器对象,可用于模拟HTTP的GET和POST请求。当一个网页需要和服务端进行异步通信时,客户端浏览器创建XMLHttpRequest,并设置参数,指定服务器端的交互页面和处理服务器返回的函数,完成两个网页之间的异步通信。具体的网页更新工作可通过当前页面的DOM模型完成,配合JavaScript可以实现页面数据在无刷新下的定时数据更新,在聊天室、文字直播上有良好的页面效果。

2. 创建XMLHttpRequest对象

在使用XMLHttpRequest对象之前,必须先创建一个XMLHttpRequest对象。由于XMLHttpRequest不是一个W3C标准,不同的浏览器对XMLHttp的支持不同,主要表现

在创建 XMLHttpRequest 对象实例的方法不同。IE 浏览器把 XMLHttpRequest 实现为一个 ActiveX 对象，其他浏览器（如 FireFox、Safari 和 Opera）把它实现为一个本地 JavaScript 对象。由于存在这些差别，JavaScript 代码中必须包含有关的逻辑，从而使用 ActiveX 技术或者使用本地 JavaScript 对象技术来创建 XMLHttpRequest 的一个实例。

（1）IE 浏览器中创建 XMLHttpRequest 对象

在 IE 浏览器中，使用 ActiveXObject 创建 XMLHttpRequest 对象，一般形式是：

```
Varhttp_request = new ActiveXObject("Microsoft.XMLHTTP");
```

其中，Microsoft.XMLHTTP 为 ProgID。在 Windows 操作系统中，存在过多个类似的 ProgID，包括 Microsoft.XMLHTTP、Msxml2.ServerXMLHTTP、Msxml2.XMLHTTP 等。简单地讲，这些不同的 ProgID 与 IE 浏览器的版本有关。究竟应该指定怎样的 ProgID 呢？可以使用下列代码来检测浏览器支持的 ProgID：

```
var progIDs = ["Msxml2.XMLHTTP.6.0", "Msxml2.XMLHTTP.5.0", "Msxml2.XMLHTTP.4.0",
"Msxml2.XMLHTTP.3.0", "Msxml2.XMLHTTP","Microsoft.XMLHTTP"];
for (var i = 0; i < progIDs.length; i++) {
   try {
        var http_request = new ActiveXObject(progIDs[i]);
        return http_request;
   }
   catch (ex) {}
}
return null;
}
```

上述代码从最新的版本开始检查，只要找到一个系统中存在的版本即可退出。对于这些不同的版本，并不都是为了 IE 浏览器安装的，有的需要单独安装，有的随 MS Office，MS SQL Server 一并安装。对于 IE 浏览器而言，Msxml2.XMLHTTP 是系统自带的组件，其他版本的 Msxml2.XMLHTTP 并不一定在系统中存在。

根据上述说明，在 IE 浏览器中，使用 ActiveXObject 方式创建 XmlHttp 对象，代码如下：

```
if (window.ActiveXObject)
{
  try{
     http_request = new ActiveXObject("Msxml2.XMLHTTP");
  }
  catch(e){
    try{
     http_request = new ActiveXObject("Microsoft.XMLHTTP");
    }
    catch(e) { alert('浏览器不支持 ajax'); }
   }
}
```

（2）其他浏览器中创建 XMLHttpRequest 对象

除了 IE 浏览器，其他浏览器采用本地 JavaScript 对象创建 XMLHttpRequest 对象，这也是 W3C 标准规定的创建 XMLHttpRequest 对象的方法。一般形式如下：

```
http_request = new XMLHttpRequest();
```

在创建 XMLHttpRequest 对象的方法上,IE 浏览器不同于其他浏览器。后来微软开始意识到这个问题给微软带来的不利影响,在 IE 7 以后的版本中,开始使用 W3C 标准规定的方法创建 XMLHttpRequest 对象。也就是说,对微软 IE 7 之后的版本和非 IE 浏览器,创建 XMLHttpRequest 对象方法是一样的,从此,XMLHttpRequest 对象的创建方法开始走向统一。

根据以上分析,要保证 AJAX 页面在不同的浏览器中功能正常,需要根据不同的浏览器选择创建 XMLHttpRequest 对象的不同方法。对浏览器类型的区分,既可以通过浏览器对象 navigator 来实现,也可以通过检查浏览器是否提供对 XMLHttpRequest 对象和 ActiveX 对象的支持来实现。如果浏览器支持 ActiveX 对象,就可以使用 ActiveX 来创建 XMLHttpRequest 对象。否则,就要使用本地 JavaScript 对象技术来创建。

创建具有跨浏览器功能的 XMLHttpRequest 对象函数代码如下:

```
function createXMLHttpRequest()
{
    var http_request = null;
    if (window.XMLHttpRequest){            //IE 7 及更高版本,其他浏览器
      http_request = new XMLHttpRequest();
    }else if (window.ActiveXObject){       //IE 7 以前的版本
        http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

目前,对于市场上的主流浏览器,已经全面支持 XMLHttpRequest 对象,在创建对象的方法上,IE 浏览器和其他浏览器已经统一,这为保证页面的跨浏览器支持提供了方便。

3. XMLHttpRequest 对象属性和方法

XMLHttpRequest 对象常用属性及方法见表 5-29。

表 5-29 XMLHttpRequest 常用属性及方法

常 用 属 性	
onreadystatechange	状态改变时发生的事件,可以将它与一个 JavaScript 函数绑定
readyState	请求状态。有 5 个可能的取值(0:未初始化,1:正在加载,2:已加载,3:交互中,4 :完成)
responseText	服务器响应,表示为一个串
responseXML	服务器响应,表示为 XML。可以解析为一个 DOM 对象
status	服务器 HTTP 状态码。例如 200:OK,404:Not Found,等等
statusText	HTTP 状态码相应文本。例如 OK 或 Not Found(未找到)等
常 用 方 法	
open(method, url, asynch)	建立对服务器的请求,即设置对应服务器端的异步通信程序,URL 为服务器端程序的 URL
send(content)	向服务器发送请求
setRequestHeader(header, value)	把指定首部设置为所提供的值,在设置任何首部之前必须先调用 open()
abort()	停止当前请求
getAllResponseHeaders()	把 HTTP 请求的所有响应首部作为键/值对返回
getResponseHeader(header)	返回指定首部的串值

下面介绍这几个方法的应用：

(1) void open(string method, string url,boolean asynch[,string username][, string pwd])

建立对服务器的调用，这是初始化一个请求的纯脚本方法，参数说明如下：

method：必选参数，提供调用的特定方法(GET、POST 或 PUT)。

url：必选参数，所调用资源的 URL，即服务器端处理程序。

asynch：必选参数，指示这个调用是异步的还是同步的。默认值为 true，如果为 false，处理就会等待，直到从服务器返回响应为止。

username：可选参数，表示用户名，用于身份验证。

pwd：可选参数，表示用户密码，用于身份验证。

(2) void send([content])

向服务器发出请求。如果请求声明为异步的，这个方法就会立即返回，否则它会等待直到接收到响应为止。可选参数可以是 DOM 对象的实例、输入流，或者串。传入这个方法的内容会作为请求体的一部分发送。

(3) void setRequestHeader(string header, string value)

HTTP 请求中一个给定的首部设置值。它有两个参数：header 表示要设置的首部；value 表示首部的值。需要说明，这个方法必须在调用 open()之后才能调用。

(4) void abort()

停止请求的方法。

(5) string getAllResponseHeaders()

返回所有响应的 HTTP 头，首部包括 Content-Length、Date 和 URI。

(6) string getResponseHeader(string header)

返回指定的首部值。

【例 5-16】 设计一个用户注册的页面，使用 AJAX 技术完成用户账户的检测。

分析：用户账户注册是 Web 开发中最常用的功能，在填写表单详细信息以前，检测申请的账户是否存在，可以避免提交表单后，发现账户已经存在而使得用户重新填写。

设注册页面(局部)设计如图 5-16 所示。

图 5-16 用户注册页面(部分)

用户注册页面涉及 4 个文档：①注册页面 user-add.htm。②用户名检测页面 user-checkaccount.jsp。③验证码显示 verificationcode.jsp。④用户信息保存页面 user-addsave.jsp。后三个页面均为服务器页，有关 AJAX 的是注册页面。相关代码如下：

(1) 注册页面 user-add.htm 代码清单如下：

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

```
<head>
<meta charset = "utf - 8">
<script type = "text/javascript">
////////////////////////////////////////////////////////////////////////////////
function PromptAccountInput()
{
    RtnAreaID.innerHTML = "用户账户是由字母、数字构成的,不能包含汉字字符";
}
function PromptAccountCheck()
{
    RtnAreaID.innerHTML = "<a href = '#' onclick = 'AccountCheck(form1.useraccount)'>-- 检测
用户账户 --</a>";
}
////////////////////////////////////////////////////////////////////////////////
function AccountCheck(obj)
{
  var useraccount = obj.value;
  if(useraccount == "")
  {
    window.alert("用户账户不能为空!");
    obj.focus();
    return false;
  }
  if (escape(form1.useraccount.value).indexOf("%u")!= - 1)
  {
    alert("用户账户不能包含汉字,请重新输入");
    obj.focus();
    return false;
  }
  //6 - 8 位的字母、数字、下画线、句点、@,组成,不能包含汉字
  var Expression = /^[A - Za - z]{1}([A - Za - z0 - 9]|[._@]){5,8}$/;
  var objExp = new RegExp(Expression);
  if (objExp.test(useraccount) == false)
  {
    alert("用户账户由 6 - 8 位字母、数字、点、下画线、@组成,首字符为字母.");
    obj.focus();
    return false;
   }
  RtnAreaID.innerHTML = "请等待……";
  //初始化对象并发出 XMLHttpRequest 请求
  http_request = false;
  //IE 7 以后版本,或其他浏览器
  if (window.XMLHttpRequest)
  {
     http_request = new XMLHttpRequest();
  }
  else //IE7 以前版本
  if (window.ActiveXObject)
  {
    try{
       http_request = new ActiveXObject("Msxml2.XMLHTTP");
```

```
    }
    catch(e){
      try{
       http_request = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch(e) { alert('浏览器不支持 ajax'); }
     }
  }
  if (!http_request)
  {
    alert("不能创建 XMLHTTP 实例!");
    return false;
  }
  //指定处理服务端返回数据的函数
  http_request.onreadystatechange = CheckReturn;
  //发出 HTTP 请求
  http_request.open("GET", "newuser - accountcheck.jsp?useraccount = " + useraccount,true);
  http_request.send(null);
}
////////////////////////////////////////////////////////////////////////////////
//处理服务器返回的信息,即 JSP 页面的输出
function CheckReturn()
{
   if (http_request.readyState == 4)
   {
    if (http_request.status == 200)
       RtnAreaID.innerHTML = http_request.responseText;
    else
       window.alert(http_request.status);
   }
}
////////////////////////////////////////////////////////////////////////////////
//重载验证码
function VerifyCodeNew(obj)
{
   var timenow = new Date().getTime();
   obj.src = "vcode - create.jsp?d = " + timenow;
}
</script>
</head>
<body>
<form name = "form1" method = "post" action = "user - addsave.jsp">
<table>
<tr>
   <td width = "20 %"><span class = "red">*</span>用户账户:</td>
   <td width = "40 %">
       <input type = "text" name = "useraccount" onfocus = "PromptAccountInput()"
onblur = "PromptAccountCheck()"/>
   </td>
   <td id = "RtnAreaID">
       <a href = '#' onClick = 'AccountCheck(form1.useraccount)'>-- 检测用户账户 --</a>
```

```
    </td>
</tr>
<tr>
<td>验证码:</td>
<td>
  <input type="text" name="myverifycode" value=""/>
  <img src="vcode-create.jsp" width="60px" height="22px" id="vcode">
  <a href="VerifyCodeNew(document.getElementById('vcode'))">看不清</a>
</td>
<td>请输入验证码,区分大小写</td>
</tr>
</table>
</form>
</body>
</html>
```

对于 AJAX,比较麻烦的是创建 XMLHttpRequest 对象,早期的 IE 和其他浏览器不同,现在已经统一。上述代码在微软的 IE 11 浏览器和 Google Chrome 浏览器中调试通过。

(2) 用户名检测页面 user-checkaccount.jsp,代码清单如下:

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />
<% //接收客户端提交的数据
String useraccount=request.getParameter("useraccount");
if (useraccount==null) useraccount="";
ResultSet rs = null;
try{
     String strSQL = "SELECT UserAccount FROM useraccounts
                    WHERE UserAccount='" + useraccount + "'";
     rs=gslpub.executeQuery(strSQL);
     if (rs.next())
       out.println("<font color='red'>抱歉!账户" + useraccount + "已经被注册</font>");
     else
       out.println("<font color='blue'>祝贺您!账户" + useraccount + "可以使用</font>");
}
catch (Exception ex){
     out.print(ex.getMessage());
}
finally {
     gslpub.disconnectToDB();
}
%>
```

在上述服务端脚本程序中,首先读取客户端传递的参数 useraccount,在数据库中查找,根据查找结果输出数据。该输出被返回到客户端,被显示在客户端页面指定的 id 区域。

5.9 JavaScript 库

在 JavaScript 客户端脚本语言出现后,在 JavaScript 中,虽然提供了一组标准的内置对象、浏览器对象和 DOM 对象,但这些对象的功能有限,许多功能实现依然需要大量的用户

编码。在 JavaScript 语言基础上，一大批 JavaScript 编程高手开始积极地研发 JavaScript 程序库，以扩展 JavaScript 的功能，从而提高开发人员的编程效率。

5.9.1 库与框架

2005 年，Web 开发者 Sam Stephenson 开发了第一代 JavaScript 程序库 Prototype.js，这是一个 JavaScript 开发基础类库。在当时浏览器创新一片死气沉沉的景象中，Prototype 是一个与众不同的创意：我们能否通过扩充 JavaScript 的内置类型、通过增加具有新功能的类型来弥补 JavaScript 的固有缺陷？随后这种思想被广泛接受，在许多著名的网站出现了 Prototype 的身影。

然而，不久之后，人们看到，Prototype 的核心思想和 JavaScript 的发展方向是不一致的。因为，浏览器厂商对 JavaScript 所做的努力是增加新的 API，其中很多与 Prototype 的实现相冲突。此时，程序员展现出对一些小的、自我实现、模块化的脚本库的偏爱，而不是大型的框架，Prototype 表现出了架构上的缺陷，逐渐淡出舞台。

尽管如此，Prototype 曾给众多程序员带来帮助，它对后来很多 JavaScript 库的研发产生过重大影响，Prototype 对前端技术进步做出的贡献不可磨灭。在软件的世界里，人们总是在尝试新的思想，纠正过去的不足。

2006 年 1 月，美国人 John Resig 提议改进 Prototype 的“Behaviour”库，他在 Blog 上发表了自己的想法并给出具体的例子说明。随后，John Resig 在纽约 barcamp 发布了一个新的 JavaScript 程序库，即 jQuery。顾名思义，jQuery 是 JavaScript 和 Query(查询)的意思，是一个辅助 JavaScript 开发的程序库。

2006 年 8 月，该库第一个稳定版本 jQuery 1.0 发布，具有对 CSS 选择符、事件处理和 AJAX 交互的稳健支持。随后 jQuery 团队再接再厉，对 jQuery 进行持续的优化和改进，先后发布了 jQuery 1.0，jQuery 1.1，…，jQuery 1.8，jQuery 2.0(2013 年 3 月)，jQuery 2.1(2014 年 4 月)到 jQuery3.0(2016 年 6 月)一系列版本，成为最受 JavaScript 开发人员欢迎的 JavaScript 程序库。

简单地讲，jQuery 是一个 JavaScript 框架，使用类似于 CSS 选择器，可以快速找到文档中的 HTML 元素，并对其进行操作，如隐藏、显示、改变样式、添加内容等。能够方便地在页面上添加和移除 HTML 元素。这些功能虽然使用 JavaScript 也能实现，但是 jQuery 使这些工作变得更加简单。

5.9.2 jQuery 基础

和传统程序设计语言的函数库、类库不同，作为 JavaScript 脚本语言的程序库，jQuery 是以源码的形式提供的。jQuery 是一个 JavaScript 函数库，保存为一个 JavaScript 文件(扩展名为.js)，其中包含了所有的 jQuery 函数。要使用 jQuery 库函数，需要下载相应的 JS 文件保存到本地服务器直接引用，也可以从多个公共服务器 CDN 中引用。

1. 使用 jQuery 库

为使用方便，通常将 jQuery 库下载并保存到 Web 服务器中。作为开源软件，可以登录 jQuery 官方网站(http://jquery.com/)免费下载，包括非压缩版和压缩版。在本书中，我们下载的是压缩的开发版 jquery-3.1.1.min.js。虽然都是文本文件，但压缩文件经过了特殊

处理,虽然可以使用 SublimeText 代码编辑器打开 jquery.min.js 文件,但很难阅读。

除了将 jQuery 下载到 Web 服务器上外,还可以借助内容分发网络(Content Delivery Network,CDN)[①]来使用 jQuery 库。把 jQuery 存储在 CDN 公共库上可加快网站载入速度,CDN 公共库是指将常用的 JS 库存放在 CDN 节点,以方便广大开发者直接调用。与将 JS 库存放在服务器单机上相比,CDN 公共库更加稳定、高速,且可以提供最新版本。为 jQuery 提供 CDN 服务的包括:Google、Microsof,新浪云计算(SAE)、百度云(BAE)等。

每一个 CDN,都提供了稳定的 jQuery 包含网址,下面是几个常用的 CDN 网址:

Google CDN:

```
<script type="text/javascript"
      src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.0/jquery.min.js">
</script>
```

Microsoft CDN:

```
<script type="text/javascript"
      src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.0.min.js">
</script>
```

新浪 CDN:

```
<script type="text/javascript"src="http://libs.baidu.com/jquery/2.0.3/jquery.min.js">
</script>
```

百度 CDN:

```
<script type="text/javascript"src="http://libs.baidu.com/jquery/2.0.3/jquery.min.js">
</script>
```

为保险起见,当无法从 CDN 服务器上获取 jQuery 时,则使用本地 jQuery,需要在网页的< head >部分包含 jQuery 库文件。一般形式如下:

```
<script type="text/javascript" src="/jquery/jquery.min.js"></script>
```

在具体引用 jQuery 的代码中,版本号以实际安装的 jQuery 库文件为准。在 JS 文件命名中,一般使用减号(-)表示语义中的空格,使用点(.)表达从属关系,经过压缩的源文件通常使用“min”表示,区别于原始版本。例如:jquery.min.js 表示 jQuery 库压缩文件。所谓 JavaScript 压缩文件,就是利用专门工具,将文件进行去冗处理、替换函数内部变量等操作,最大程度减少文件长度。JavaScript 文件压缩后,JavaScript 解释机可以理解,但用户将不再可读,对程序起到一定的保护作用,这是保护脚本程序的一种常用方法。

2. jQuery 的组成

在 jQuery 库文件中,包含了大量的库函数,可以将 jQuery 库函数分成以下类型:

---

① 内容分发网络 CDN,基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节,使内容传输得更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络。CDN 系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容,解决 Internet 网络拥挤的状况,提高用户访问网站的响应速度。

(1) HTML 元素选取；
(2) HTML 元素操作；
(3) DOM 遍历和修改；
(4) CSS 操作；
(5) HTML 事件函数；
(6) JavaScript 特效和动画；
(7) AJAX；
(8) Utilities。

在 Web 前端开发中，与传统的 JavaScript 原始编码相比，使用 jQuery 库更加简便高效，且 jQuery 兼容多种浏览器，jQuery 编程的优势主要体现在以下 4 个方面：①HTML 文档遍历操作；②事件处理；③动画；④AJAX 互操作。本质上讲，jQuery 也是 JavaScript，只是把前端开发中大量的公共功能进行了封装，构成了一个 JavaScript 开发框架。

5.9.3 jQuery 函数

“写得更少，做得更多”是 jQuery 的基本出发点。这一思想是通过丰富的 jQuery 函数库实现的。本质上讲，jQuery 函数也是用 JavaScript 程序写成的。但是，jQuery 为简化用户编码，对 HTML 文档的常用操作进行了抽象和封装，写成了一组可供用户直接调用的函数。

1. 元素选择器

在 JavaScript 中，选取元素通常通过 document 对象成员函数和成员变量来实现，例如 document.getElementByID()、getElementsByName()和 getElementsByTagName()函数可以返回指定的文档对象。也可以通过 document 对象的 all、images、links、forms 等数组成员变量返回相应的文档对象。这种获得 HTML 元素对象的代码可读性强，但书写比较麻烦。

在 jQuery 中，选取 HTML 元素，是通过 jQuery 选择器完成的。jQuery 选择器包括元素选择器和属性选择器，可通过标签名、属性名或内容对 HTML 元素进行选择，可以选择一个元素，也可以选取一个元素数组。一般形式是：

```
$(selector)
```

其中，符号 $ 表示 jQuery，选择符(selector)用于“查询”和“查找”要返回的 HTML 元素，返回的元素可能是一个，也可能是多个。

(1) 使用 CSS 选择器选取 HTML 元素，例如：$("p")选取所有< p >元素，$("p.note")选取所有 class="note"的< p >元素，$("p#help")选取 id="help"的第一个< p >元素。

(2) 使用 XPath 表达式选择带有给定属性的元素，例如：$("[href]")选取所有带有 href 属性的元素，$("[href='#']")选取所有带有 href 值等于"#"的元素。$("[href!='#']")选取所有带有 href 值不等于"#"的元素。$("[href$='.jpg']")选取所有 href 值以".jpg"结尾的元素。

当选择了一个或多个 DOM 对象后，可以对选取的对象进行操作，例如：

$("p").hide()：隐藏所有段落。

$("p.test").hide()：隐藏所有 class="test"的段落。

$("#test").hide()：隐藏所有 id="test"的元素。

$("p").css("background-color","gray")：设置所有 p 元素的 CSS 属性值。

2. 元素操作函数

对于选取的 HTML 元素，可以对元素内容和属性进行设置，jQuery 提供的元素内容和属性操作函数见表 5-30。

表 5-30 jQuery 元素操作函数

函 数 名	功 能
$(selector).html(content)	设置被选元素的 innerHTML 值
$(selector).append(content)	在被选元素的内部 innerHTML 追加内容
$(selector).prepend(content)	预置被选元素的内部 innerHTML 内容
$(selector).after(content)	在被选元素之后添加 HTML 内容
$(selector).before(content)	在被选元素之前添加 HTML 内容
$(selector).css(name,value)	为匹配元素设置样式属性的值
$(selector).css({properties})	为匹配元素设置多个样式属性
$(selector).css(name)	获得第一个匹配元素的样式属性值
$(selector).height(value)	设置匹配元素的高度
$(selector).width(value)	设置匹配元素的宽度

例如：$("p").css({"background-color":"red","font-size":"200%"})；设置所有 p 段落的背景色和字体大小。$(this).css("font-size")；返回当前元素的 font-size 属性值。

3. 效果函数

关于 HTML 中元素的隐藏、显示、切换、滑动以及自定义动画等效果，jQuery 提供的效果函数见表 5-31。

表 5-31 jQuery 效果函数

函 数 名	功 能
$(selector).hide(speed,callback)	隐藏被选元素
$(selector).show(speed,callback)	显示被选元素
$(selector).toggle(speed,callback)	切换(在隐藏与显示之间)被选元素
$(selector).slideDown(speed,callback)	向下滑动(显示)被选元素
$(selector).slideUp(speed,callback)	向上滑动(隐藏)被选元素
$(selector).slideToggle(speed,callback)	对被选元素切换向上滑动和向下滑动
$(selector).fadeIn(speed,callback)	淡入被选元素
$(selector).fadeOut(speed,callback)	淡出被选元素
$(selector).fadeTo(speed,callback)	把被选元素淡出为给定的不透明度
$(selector).animate()	对被选元素执行自定义动画

在 jQuery 效果函数中，参数 speed 是可选参数，规定显示或隐藏的速度，取值包括"slow"、"fast"、"normal"或毫秒。callback 是可选参数，设置在动画函数 100%完成之后被执行的函数名称。由于 JavaScript 语句是逐一执行的，可能会出现这样的情况，在动画还没有完成前，就开始执行动画后面的语句，这就可能会产生错误或页面冲突。为了避免这个情

况，可以参数的形式添加 Callback 函数。当动画 100％完成后，即调用 Callback 函数。

例如：

```
$("p").hide(1000);
alert("The paragraph is now hidden");
```

上面的代码是错误的，因为元素隐藏需要的时间是 1000ms，在元素隐藏完成前，就会执行到下面的提示语句，这是不正确的。遇到此种情况，可以增加 callback 函数：

```
$("p").hide(1000,function(){
    alert("The paragraph is now hidden");
});
```

自定义动画函数 animate 有 4 个参数：

params，必选参数，定义产生动画的 CSS 样式表，可以同时设置多个此类属性。

duration，可选参数，定义用来应用到动画的时间，取值是"slow"、"fast"、"normal"或毫秒。

4. 事件处理函数

在 HTML 中，我们知道标记属性包括一般属性和事件属性，事件处理函数就是指发生某些事件时所调用的方法，即对应标记的事件属性。

在 jQuery 中，事件处理函数通常写到文档头部，例如：

```
<html>
<head>
<script type="text/javascript" src="jquery/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>
<body>
<h1>Heading</h1>
<p>Paragraph1</p>
<p>Paragraph2</p>
<button>Click Here</button>
</body>
</html>
```

在上面的例子中，定义了<button>元素的 click 事件处理函数，函数定义如下：

```
$("button").click(function() {
  $("p").hide(); //函数代码部分
} );
```

当单击相应按钮时，事件被触发，调用该函数，该方法隐藏所有<p>元素。除了鼠标单击事件外，常用的事件函数还有 dblclick（鼠标双击）、focus（获得输入焦点）、mouseover（鼠标悬停）等。

在上面的例子中,我们定义的所有 jQuery 函数都包含在一个 document.ready(文档就绪函数)函数中,形式如下:

```
$(document).ready(function(){
    --- jQuery functions go here ----
});
```

这是为了防止文档在完全加载(就绪)之前运行 jQuery 代码。如果在文档没有完全加载之前就运行函数,操作可能失败。例如:试图隐藏一个不存在的元素,获得未完全加载的图像的大小等。

在 Web 页面较多时,我们可以将那些公共的 jQuery 函数保存为一个单独的.js 文件,把这些文件保存在一个单独的文件夹中。在需要的网页中,包含相应的.js 文件,例如:

```
<script type="text/javascript" src="pubjs/treemenudata.js"></script>
```

使用单独的.js 文件,需要注意的是一定要避免不同文件中,函数等名称的冲突问题。

5. AJAX 函数

jQuery 提供了用于 AJAX 开发的丰富函数(方法)库,见表 5-32。

表 5-32 AJAX Request 函数

函 数 名	功 能
$.ajax(options)	把远程数据加载到 XMLHttpRequest 对象中
$(selector).load(url[,data][,callback])	把远程 HTML 数据加载到被选的元素中
$.get(url[,data][,callback][,type])	使用 HTTP GET 加载远程数据
$.post(url,data,callback,type)	使用 HTTP POST 加载远程数据
$.getJSON(url,data,callback)	使用 HTTP GET 加载远程 JSON 数据
$.getScript(url,callback)	加载并执行远程的 JavaScript 文件

函数参数说明如下:

- boptions,完整 AJAX 请求的所有的参数项,包括 url、type、async、contentType、data、dataType、beforeSend、complete、error、success。其中,anysc 参数默认值为 true,默认设置下,所有请求均为异步请求,如果需要发送同步请求,将此选项设置为 false。同步请求将锁住浏览器,用户其他操作必须等待请求完成才可以执行。

参数 beforeSend、complete、error、success 均为函数参数,代表发送请求前、完成后、失败和成功下调用的函数,complete 函数不管是失败和成功均调用。success 函数有两个参数,一个为由服务器返回,并根据 dataType 参数进行处理后的数据,一个为描述状态的字符串,该参数可选。

- burl,被加载的数据的 URL 地址。
- bdata,发送到服务器的数据的键/值对象。
- bcallback,回调函数,当数据被加载时,所执行的函数,例如 $("#showget").html(req);,在特定的位置显示服务端返回的数据。
- btype,被返回的数据的类型,取值为 html、xml、json、jasonp、script、text。

下面是一个典型的 AJAX 函数:

```
$('#login').click(function(){
  $.ajax({
    type:"GET",
    url:"login.jsp",
    data:{useraccount:$("#useraccount").val(),pwd:$("#pwd").val()},

    success:function(rtndata){
      $('#resText').empty();
      $('#resText').html(rtndata);
    }
  });
});
```

最后需要说明的是，由于本书篇幅所限，不能全面介绍 jQuery 函数库，对 jQuery 中更多的 jQuery 函数、jQuery 事件、jQuery 效果、jQuery 文档方法、jQuery 遍历函数、jQuery 数据操作函数等，请读者查阅 jQuery 相关参考手册。

5.9.4 jQuery 插件

在 jQuery 中，不仅有丰富的 jQuery 函数，同时还有许多成熟的插件可供选择。所谓 jQuery 插件，可以简单地理解为具有特定功能和用途的 jQuery 功能模块，它是 jQuery 函数的延伸，利用 jQuery 而编写的一个功能模块。每个插件保存为一个独立的.js 文件，在网页中使用插件，必须要包含该文件。

在 jQuery 之上，Web 开发人员研发了数量众多的 jQuery 插件，这些插件数量众多，按照功能分类，常用的插件如下。

(1) 菜单类插件：水平菜单，垂直菜单，树状菜单，下拉菜单，右键菜单。

(2) 日期时间插件：用于输入日期和时间，插件很多，根据需要选用。

(3) 表单验证插件。

(4) 数据组织插件：列表插件，表格排序插件，网格插件，电子表格插件等。

(5) 图形绘制插件。

(6) 图片插件：图片裁剪插件，图片上传插件。

(7) 地图插件。

(8) 文件上传插件，文件下载插件。

(9) 进度条插件，星际评定插件等。

除了使用他人的 jQuery 插件外，我们也可以开发自己的 jQuery 插件。jQuery 插件开发有两种方式：一种是类扩展的方式开发插件，为 jQuery 添加新的全局函数(jQuery 的全局函数是属于 jQuery 命名空间的函数)，如果将 jQuery 看成一个类，那么就相当于给 jQuery 类本身添加方法。例如：

```
$.ltrim = function( str ) {
        return str.replace( /^\s+/, "" );
};
```

第二种方式是对象扩展的方式开发插件，即为 jQuery 对象添加方法。

在网页中使用 jQuery 插件，就是将插件文件包含到网页中。使用 jQuery 插件，不需要

编写大量的 JavaScript 代码,不仅减少了编程的工作量,同时还可以做到界面美观。

5.9.5 举例

使用 jQuery 库,与传统的 JavaScript 编程相比,风格上有所变化,有些程序的可读性不如传统的 JavaScript 编码。但丰富的 jQuery 函数库,可以显著提高编程效率,因此,在前端开发中,应该加强 jQuery 的使用。

下面是一个现代风格的表格 UI 设计,要求表格隔行显示不同的背景色,表头为特定颜色。在功能实现时,采用了 jQuery 函数定义风格,代码如下:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
.content-table{
    width:100%;
    border-collapse:collapse;          /*合并边框间隙*/
    empty-cells:show;                  /*显示内容为空的单元格*/
    font-size:17px;
    text-align:center;
}
.content-table tr{                     /*设置行高*/
    height:35px;
}
.content-table th{                     /*表头单元格*/
    padding:.5em;
    border:1px solid #fff;
    background:#3992d0;
    color:#fff;                        /*文本颜色*/
}
.content-table td{                     /*单元格*/
    border:1px solid #fff;
}

.content-table tr.even td{             /*表格颜色隔行显示*/
    background:#e5f1f4;                /*深色*/
}
.content-table tr.odd td{
    background:#f8fbfc;                /*浅色*/
}
</style>
<script src="jquery/jquery.min.js"></script>
<script type="text/javascript">
this.tablestyle = function(tableobj){
    var css = "odd";
    var tr = tableobj.getElementsByTagName("tr");
    for (var i=0;i<tr.length;i++){
        css = (css == "odd") ? "even" : "odd";
        tr[i].className = css;
```

```
    };
};
this.tablecloth = function(){
    var tables = document.getElementsByTagName("table");
    for (var i = 0;i < tables.length;i++){
        tablestyle(tables[i]);
    };
};
window.onload = tablecloth;
</script>
</head>
<body>
<table class = "content - table">
    <tr>
        <th width = "30 % ">学号</th>
        <th>姓名</th>
    </tr>
    <tr>
        <td> 2016100001 </td>
        <td>王一平</td>
    </tr>
    <tr>
        <td> 2016110001 </td>
        <td>岳颖</td>
    </tr>
    <tr>
        <td> 2016110002 </td>
        <td>欧阳天雨</td>
    </tr>
</table>
</body>
</html>
```

在 IE、Google Chrome 浏览器中打开网页，显示如图 5-17 所示。

图 5-17　表格的隔行背景设置

在上述代码中，我们使用了 HTML 5 的 UTF-8 字符集。在实际开发中，因为网页制作工具的不同，有的网页采用的是 GB2312 字符集，有时候简单地修改网页文件中的字符集编

码,保存网页后,用代码编辑工具打开网页,或者在浏览器中打开网页时可能会显示乱码。

在网页中显示乱码或在文本编辑器中显示乱码,本质上都是文件存储的字符编码和显示设置的字符编码不一致造成的。此时,可以修改网页中的显示字符编码设置,使其和网页文件存储的字符编码一致,文件存储字符编码是由网页编辑工具确定的。或者,将页面内容复制到 Windows 记事本程序中,修改记事本文件存储编码,使其和网页显示设置一致。

5.10 综 合 举 例

Web 客户端的编程比较简单,它不需要特别的编译和运行环境,只要有一个浏览器就可以了。但是,要编写高质量的客户端脚本,并不容易,这需要大量的实践经验,也来源于用户的需求。没有需求,就不会有深入的编程体验,更无法把一个工具学精。为此,在本章的最后,我们给出三个在实际 Web 开发中常用的功能,分别来综合说明 JavaScript 中的窗口控制、图层技术和 HTML DOM 文档操作,帮助建立总体的 Web 客户端编程思想。也相信这三个应用的代码,对用户会有很好的借鉴作用。

5.10.1 创建折叠式菜单

在许多网页上,都有折叠式菜单,这类菜单的创建可以利用 HTML DOM,通过纯 JavaScript 程序编码实现。目前常用的菜单有折叠式和树状目录结构两种形式。本小节介绍折叠式菜单的创建和应用。

设有一个框架页面,分为左右两个 frame,左侧显示一个折叠菜单,右侧是菜单项所对应的页面显示区域,右侧帧名为 mainFrame。左侧菜单页面代码如下:

折叠菜单 mainmenu.htm 代码清单:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
a {font-size:17px;text-decoration:none}
a:link{color:#0000FF;}
a:visited{color:#0000FF;}
a:hover {color:#FF0000;font-weight:bold;}
a:active {color:#0000FF;}
.menutitle {
    width:150px;height:30;
    border:1px solid #204848;
    font-size:17px;
    color:rgb(254,254,166);
    background-color:rgb(0,119,166);
    text-align:center;
    cursor:pointer;
}
</style>
<script>
function SwitchMenu(submenu)
{
```

```
  for (var i = 0;i < menutable.rows.length;i++)
  {
      if (menutable.rows[i].getAttribute("submenu") == submenu)
      {
         if (menutable.rows[i].style.display == "" || menutable.rows[i].style.display == "
block")
            menutable.rows[i].style.display = "none";
         else
           menutable.rows[i].style.display = "block";
      }
  }
}
</script>
</head>
<body>
<table id = "menutable" border = "0" bgcolor = "#CCCCCC">
<tr>
  <td class = "menutitle" onclick = "SwitchMenu('sub01')">课程简介</td>
</tr>
<tr submenu = "sub01">
  <td><a href = "kcjj/kcjj.htm" target = "contentframe">教学目标</a></td>
</tr>
<tr submenu = "sub01">
  <td><a href = "kcjj/kcdg.htm" target = "contentframe">教学大纲</a></td>
</tr>
<tr>
  <td class = "menutitle" onclick = "SwitchMenu('sub02')">教学队伍</td>
</tr>
<tr submenu = "sub02">
  <td><a href = "jxdw/hao.htm" target = "contentframe">课程负责人</a></td>
</tr>
<tr submenu = "sub02">
  <td><a href = "jxdw/jxtd.htm" target = "contentframe">教学团队</a></td>
</tr>
</table>
</body>
</html>
```

对于上述代码，分别在 IE 浏览器和 GoogleChrome 浏览器中打开，显示创建的折叠菜单结果，如图 5-18 所示。

图 5-18　折叠式菜单示例

当用户在一级菜单上单击时,将打开对应的二级菜单。二级菜单对应具体的超链接,当用户鼠标指向二级菜单项目时,在浏览器状态栏,可以看到对应的超链接。实现上述功能的代码很多,在本处给出的代码中,采用表格布局比 div 更加简单。其次,对菜单分组,在< tr >标记中添加了一个自定义属性 submenu,来实现菜单分组。

如果要增加三级菜单,可以对菜单编码进行规划,如 subxxyyzz,每级对应两位数字,如果是菜单标题,相应的子菜单部分别为 00。例如,sub010000 代表第一组菜单标题,sub010100 代表二级菜单 sub0101 的标题,只有 zz 不为 00 的才是具体的菜单命令。通过字符串函数 substring 可以控制一级、二级菜单的折叠和打开。

现在许多网站采用一级菜单水平布局,二级菜单采用下拉式,如何实现呢? 利用表格、自定义标记属性以及 CSS 技术,也可以很灵活地进行控制。通过动态地设置元素 className 属性为不同的 CSS 样式类,可以实现当鼠标移动到某菜单标题时,对菜单标题高亮显示等效果。对于菜单命令项,则通过定义< a >标记不同状态的 CSS 样式实现显示效果控制。

5.10.2 创建树状菜单

属性菜单也是 Web 开发中常用的菜单,下面我们通过图层的方式来演示树状菜单的创建,菜单分为二级和三级菜单。要实现的显示界面如图 5-19 所示。

图 5-19 树状菜单

对应上述树状菜单的 menutree.htm 代码清单如下:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style type="text/css">
    a {font-size:14px;text-decoration:none}
    a:link{color:#0000FF;}
    a:visited{color:#0000FF;}
    a:hover {color:#FF0000;font-weight:bold;}
    a:active {color:#0000FF;}
    img{vertical-align:middle;}
    .menutitle1{font-size:17px;color:#700000;font-weight:bold;}
    .menutitle2{font-size:17px;color:#005000;}
  </style>
  <script>
////////////////////////////////////////////////////////////////////////////////
//一级菜单和二级菜单的打开和折叠
function switchmenu(submenu)
{
  var len = submenu.length;
  for (var i = 0;i < menutable.rows.length;i++)
  {
    menuitem = menutable.rows[i].getAttribute("menuitem");
    //一级菜单标题
    if (menuitem == submenu + "0000") continue;
    //二级菜单标题
    if (menuitem == submenu + "00") continue;
```

```
    //包含的子菜单及菜单命令
    if (menuitem.substring(0,len) == submenu)
    {
       if (menutable.rows[i].style.display == "" || menutable.rows[i].style.display == "block")
          menutable.rows[i].style.display = "none";
       else
         menutable.rows[i].style.display = "block";
    }
  }
}
</script>
</head>
<body style = "overflow - x:hidden;">
<table id = "menutable" border = "0" bgcolor = "#CCCCCC">
<tr menuitem = "menuroot">
  <td style = 'cursor:pointer;'>
    <img src = 'images/applogo.gif'>
    <span>组织机构与人员岗位配置管理</span>
  </td>
</tr>
<tr menuitem = "Menu010000" onclick = "switchmenu('Menu01');">
  <td style = 'cursor:pointer;'>
    <img id = 'img01a' src = 'images/menuplus.gif'>
    <img id = 'img01b' src = 'images/folderclose.gif'>
    <span class = 'menutitle1'>用户账户管理</span>
  </td>
</tr>
<tr menuitem = "Menu010101" style = "display:none">
   <td>
     <img src = 'images/menuline.gif'>
     <img src = 'images/menulineminus.gif'>
     <span><a href = "modules - admin.jsp" target = "mainFrame">用户账户列表</a></span>
   </td>
</tr>
<tr menuitem = "Menu010102" style = "display:none">
  <td>
    <img src = 'images/menuline.gif'>
    <img src = 'images/menulineminus.gif'>
    <span><a href = 'modules - admin.jsp' target = "mainFrame">添加修改删除</a></span>
  </td>
</tr>
<tr menuitem = "Menu010103" style = "display:none">
  <td>
    <img src = 'images/menuline.gif'>
    <img src = 'images/menulineminus.gif'>
    <span><a href = 'modules - admin.jsp' target = "mainFrame">导入导出</a></span>
  </td>
</tr>
<tr menuitem = "Menu010200" style = "display:none" onclick = "switchmenu('Menu0102');">
  <td style = 'cursor:pointer;'>
    <img src = 'images/menuline.gif'>
```

```
        <img id='img0102a' src='images/menuplus.gif'>
        <img id='img0102b' src='images/folderclose.gif'>
        <span class='menutitle2'>查询统计</span>
      </td>
    </tr>
    <tr menuitem="Menu020000" onclick="switchmenu('Menu02');">
      <td style='cursor:pointer;'>
        <img id='img02a' src='images/menuplus.gif'>
        <img id='img02b' src='images/folderclose.gif'>
        <span class='menutitle1'>角色权限管理</span>
      </td>
    </tr>
    <tr menuitem="Menu020100" style="display:none" onclick="switchmenu('Menu0201');">
      <td style='cursor:pointer;'>
        <img src='images/menuline.gif'>
        <img id='img0201a' src='images/menuplus.gif'>
        <img id='img0201b' src='images/folderclose.gif'>
        <span class='menutitle2'>系统功能管理</span>
      </td>
    </tr>
    <tr menuitem="Menu020101" style="display:none">
      <td>
        <img src='images/line.gif'>
        <img src='images/line.gif'>
        <img src='images/line.gif'>
        <span><a href='modules-admin.jsp' target="mainFrame">功能列表</a></span>
      </td>
    </tr>
    </table>
    </body>
    </html>
```

树状菜单的实现方法很多,使用表格组织,其结构更容易理解,容易添加和删除。在表格设计的树状菜单中,一级、二级、三级菜单项都对应表格的一行,每一行可以设置一个单元格,分别显示树状节点图片和文字。菜单的打开和折叠,本质上就是表格行的显示和隐藏控制。只要为每一行设计科学的编码,行的隐藏和显示就很容易实现。

在一个单元格内,如果包含文字和图片,需要控制文字和图片在单元格内垂直居中,有两种方法:第一,在< img >中设置标记属性 align=absmiddle。第二,如果< img >很多,定义 img 选择器,即 img{vertical-align:middle;}。

通过上述代码,可以看出树状菜单的代码有很强的规律性,由于结构相似,可以编写菜单创建程序,通过定义菜单数据结构,自动生成上述代码。

5.10.3 数据有效性验证

在 Web 开发中,表单作为客户端输入数据的用户界面,为了保证数据输入的有效性,在提交表单以前,往往需要在客户端进行数据的有效性检查,这样可以有效地减少服务器的负载,提高整个 Web 系统的运行效率。

1. 表单的提交

在 HTML 中,表单提交有以下两种方法:

(1) 通过< form >的 onsubmit 事件属性,编写相应的表单输入处理函数。如果函数返回 true,则表单提交,返回 false,则表单不提交。使用该方法,表单中需要包含“提交”按钮,即 type="submit"的按钮。

(2) 通过普通按钮或超链接,通过定义 onclick 事件属性,提交表单。此时,对应的事件函数定义中,可分成两部分,前面是有效性验证,然后通过 Form 对象的 submit()方法提交,即表单名.submit()。该种方法和第一种方法相比,其程序可读性更好。

在传统程序中,通常会定义一些快捷键,也可以对表单提交定义快捷键,方法如下:

(1) 首先定义键盘操作处理函数

```
function doByKey()
{
   if ( window.event.keyCode == 113)      //F2,提交表单
     form1submit();
   if (window.event.keyCode == 27)        //Esc,放弃
     form1cancel();
}
```

(2) 在< body >标记中,添加 onkeydown 事件属性,代码如下:

```
< body onkeydown = "doByKey()">
```

2. 数据有效性验证

在表单提交前,通常需要验证数据的有效性,验证数据有效性除了采用传统的程序编码外,采用正则表达式验证数据有效性的效率更高,编程量较少。采用正则表达式验证数据的有效性,有三个步骤:①书写正则表达式;②创建正则表达式对象;③验证数据。

例如,要验证一个 Email 是否正确,可以编写下面的代码:

```
function checkemail(str)
{
//在 JavaScript 中,正则表达式使用"/"开头和结束,不使用双引号
    var Expression = /\w + ([ - . ]\w + ) * @\w + ([ - . ]\w + ) * \. \w + ([ - . ]\w + ) * /;
    var objExp = new RegExp(Expression);
    if (objExp.test(str) == true)
         return true;
    else
         return false;
}
```

在完成了一种类型数据的有效性验证的函数后,可以修改表单提交函数,使用各个表单域的有效性验证函数,即:

```
< input type = "button" value = "确定" onclick = "form1submit()">
```

表单提交函数事件 onclick 对应的函数 form1submit ()代码形式如下:

```
function form1submit ()
{
```

```
        if (myform.email.value == "")
        {
            alert("请输入 Email 地址!");
            myform.email.focus();
            return false;
    }
        if (!checkemail(myform.email.value))
        {
            alert("您输入的 Email 地址不正确!");
            myform.email.focus();
            return false;
    }
        //其他输入域验证
        form1.submit();
    }
```

当用户单击“确认”按钮时,执行用户定义的表单检查程序,检查各个输入域数据的有效性,如果数据全部有效,最后执行默认的表单处理函数,即提交表单。

本 章 小 结

本章首先介绍了程序设计的基本概念,目的是使读者从思想上对编程有一个基本的认识。然后讲解了浏览器的基本工作原理,概要介绍了 JavaScript 程序设计语言的一般语言要素,包括基本字符、数据与数据类型,表达式,语句和函数等,这是所有程序设计语言都共有的成分,也是本章 JavaScript 和下一章 JSP 服务器编程的语言基础。只有程序设计语言的语句是不能编写复杂的应用程序的,还需要一个开发环境,也就是说,需要特定的函数库或类库。因此,接下来讲解了 JavaScript 编程用到的标准库,这就是:JavaScript 内部对象及函数,浏览器对象和 HTML 文档对象,以及 Web 前端开发中使用广泛的 jQuery 程序库。最后,讲解了 Web 交互中表单输入的数据获取、数据的有效性验证、页面之间的交互等多个实例,以及多个综合例子,来讲解这些内部对象的功能和使用,所讲述的代码都来源于我们的研发项目,是 Web 开发中最具共性的内容,有很好的实用性。

习　　题

一、简答题

1. Web 浏览器的基本功能是什么? 为什么要有客户端脚本语言?

2. JavaScript 语言有哪几个组成部分? 简述各个部分的功能。

3. 在 JavaScript 中,myArray = new Array(10)是什么意思? 如何定义一个 3×4 的二维数组?

4. 什么是网页对象? 简述 window 对象和 document 对象常用的属性和方法。

5. 画出 HTML DOM 对象层次图。文档对象 document 有哪些常用的属性和方法?

6. 什么是 IntelliSense 技术? 如何知道一个 html 标记或一个 JavaScript 内部对象有哪些属性或方法?

7. 有哪些常用的浏览器事件？它们是如何触发的？

8. 什么是正则表达式？写一个正则表达式，实现验证密码是否由 6～8 位的字母、数字、下画线和英文句点(.)组成的正则表达式。

9. 对于表单输入，编写脚本程序，完成下列功能：

(1) 不提交表单，检测输入密码是否一致。

(2) 通过单击一个复选框，来实现一组复选框的全选或取消操作。

(3) 编写检查 Email 输入框输入的 Email 格式是否正确的函数。

(4) 编写检查电话输入框输入的电话号码是否正确的函数。

10. 在表单输入中，如何设置 onKeyPresss 事件属性，使得按回车键后，使输入焦点移向下一个表单元素，而不是提交表单？

11. 有一个网页 a.htm，包含一个超链接< a href= "b.htm? username=hao&account=111">，在网页 b.htm 中写一个函数获取传入的参数及参数值。

12. 什么是网页对话框？使用网页对话框，打开一个便条输入页面，将输入内容在父窗口中列表显示。

13. 在站点首页中，包含一个 login 表单，包含用户账户、密码输入框和“登录”按钮，利用 jQuery 和 AJAX 技术，编写登录验证用户账户和密码的函数，设服务端程序为 login.jsp。

二、阅读理解题

1. 阅读下面代码，写出运行结果。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<select name="sel"onchange="document.write(document.all.sel.value);"></select>
<script type="text/javascript">
var counts=0;
var arr = new Array("text1","text2","text3","text4");
counts=arr.length;
for (i=0;i<counts; i++)
{
    document.all.sel.options[i] = new Option(arr[i],("val"+i));
}
</script>
</body>
</html>
```

2. 说明下列函数的功能。

```
//参数 str 为一个字符串
function convert(str)
{
   if (str!="")
   {
     str=str.replace(/\r/g,"<br>");
     str=str.replace(/\s/g," ");
```

```
        }
        return str;
    }
```


三、编程题

1. 利用 div,设计一个总是置于页面右上角的图片广告,即当用户单击垂直滚动条时,图片一直在浏览器窗口的右上角。

2. 编写一个程序,在客户区的中央显示一个 300×200 大小的 div,并且可以用鼠标拖动。

3. 编写代码,当用户在浏览器窗口单击时,在屏幕右下角弹出一个窗口,显示单击位置的(x,y)坐标值。

4. 编写一个将网页中的表格导出为 Excel 表,并打印的程序。

5. 编写一个利用 CSS 样式实现打印页面指定内容和分页打印功能的程序。