第 5 章　JavaScript

5.1　JavaScript 概述

JavaScript 是一种用于 Web 程序开发的编程语言，它功能强大，主要用于开发交互式 Web 页面。JavaScript 不需要进行编译，可以直接嵌入 HTML 页面中，把静态页面转变成支持用户交互并响应事件的动态页面。

在浏览网页时，除了能看到静态的文本、图像，有时也能看到浮动的动画、信息框以及动态变换的时钟信息等。页面上这些实时的、动态的、可交互的网页效果在 Web 应用开发时可以使用 JavaScript 语言编写实现。下面针对 JavaScript 的起源、主要特点以及应用进行详细讲解。

5.1.1　JavaScript 的起源

JavaScript 是 Web 页面中的一种脚本编程语言，也是一种通用的、跨平台的、基于对象和事件驱动并具有安全性的脚本语言。JavaScript 的前身是 LiveScript，是由 Netscape（网景）公司开发的脚本语言。后来在 Sun 公司推出著名的 Java 语言之后，Netscape 公司和 Sun 公司于 1995 年一起重新设计了 LiveScript，并把它改名为 JavaScript。

在概念和设计方面，Java 和 JavaScript 是两种完全不同的语言。Java 是面向对象的程序设计语言，用于开发企业级应用程序，而 JavaScript 是在浏览器中执行，用于开发客户端浏览器的应用程序，能够实现用户与浏览器的动态交互。

5.1.2　JavaScript 的主要特点

JavaScript 是一种基于对象（Object）和事件驱动（Event Driven）并具有安全性能的解释性脚本语言，它具有以下主要特点：

- 解释性。JavaScript 不同于一些编译性程序语言（如 C、C++ 等），它是一种解释性程序语言，它的源代码不需要进行编译，而是直接在浏览器中解释执行。
- 基于对象。JavaScript 是一种基于对象的语言，它的许多功能来自于脚本环境中对象的方法与脚本的相互作用。在 JavaScript 中，既可以使用预定义对象，也可以使用自定义对象。
- 事件驱动。JavaScript 可以直接对用户或客户的输入做出响应，无须经过 Web 服

务程序，而是以事件驱动的方式进行的。例如单击鼠标、移动窗口、选择菜单等事件发生后，可以引起事件的响应。

- 跨平台性。在 HTML 页面中，JavaScript 依赖于浏览器本身，与操作环境无关。只要在计算机上安装了支持 JavaScript 的浏览器，程序就可以正确执行。
- 安全性。JavaScript 是一种安全性语言，它不允许访问本地硬盘，也不能对网络文档进行修改和删除，而只能通过浏览器实现信息浏览或动态交互。

5.1.3 JavaScript 的应用

作为一门独立的编程语言，JavaScript 可以做很多事情，但它最主流的应用还是在 Web 上，例如创建网页特效。使用 JavaScript 脚本语言实现的动态页面在网页上随处可见。下面介绍 JavaScript 的常见应用。

1. 验证用户输入的内容

使用 JavaScript 脚本语言可以在客户端对用户输入的内容进行验证。例如，在用户注册页面，要求用户输入注册信息，使用 JavaScript 可以判断用户输入的手机号、昵称及密码是否正确，如图 5-1 所示。如果用户在注册信息文本框中输入的信息不符合注册要求，或“确认密码”与“密码”文本框中输入的信息不同，将弹出相应的提示信息，如图 5-2 所示。

图 5-1 用户注册页面

图 5-2 弹出提示信息

2. 网页动画效果

在浏览网页时，经常会看到一些动画效果，使页面显得更加活泼、生动。使用 JavaScript 脚本语言也可以实现动画效果，例如可以在页面中实现焦点图切换效果。

3. 窗口的应用

在打开网页时，经常会看到一些浮动的广告窗口，这些广告窗口是网站最大的盈利手段。广告窗口也可以通过 JavaScript 脚本语言实现。

4. 文字特效

使用 JavaScript 脚本语言可以使文字出现多种特效,例如文字跳动等。

5.2 JavaScript 引入方式

在 HTML 文档中,JavaScript 脚本文件的使用和 CSS 样式文件类似。在 HTML 文档中引入 JavaScript 文件有两种方式:一种是在 HTML 文档中直接嵌入 JavaScript 脚本,另一种是链接外部的 JavaScript 脚本文件。

5.2.1 在 HTML 页面中嵌入 JavaScript 脚本

在 HTML 文档中,通过<script>标记及其相关属性可以引入 JavaScript 代码。<script>标记的常用属性如表 5-1 所示。

表 5-1 <script>标记的常用属性及说明

属 性	说 明
language	设置所使用的脚本语言及版本
src	设置外部脚本文件的路径位置
type	设置所使用的脚本语言,此属性已代替 language 属性
defer	当 HTML 文档加载完毕后再执行脚本语言

1. language 属性

language 属性用于指定在 HTML 中使用的脚本语言及其版本。language 属性的使用格式如下:

```
<script language="javascript"></script>
```

2. src 属性

src 属性用于指定外部脚本文件的路径。外部脚本文件通常使用 JavaScript 脚本,其扩展名为.js。src 属性的使用格式如下:

```
<script src="01.js"></script>
```

3. type 属性

type 属性用于指定 HTML 中使用的脚本语言及其版本。该属性从 HTML 4.0 标准开始,推荐使用 type 属性代替 language 属性。type 属性的使用格式如下:

```
<script type="text/javascript"></script>
```

4. defer 属性

defer 属性的作用是当文档加载完毕后再执行脚本。当脚本语言不需要立即运行时,

设置 defer 属性，浏览器将不必等待脚本语言装载，这样页面加载速度会更快。但当一些脚本需要在页面加载过程中或加载完成后立即执行时，就不需要使用 defer 属性。defer 属性的使用格式如下：

```
<script defer></script>
```

在 HTML 文档中，可以通过＜script＞标记嵌入 JavaScript 代码，具体代码如下：

```
<script type="text/javascript">
    javascript 代码
</script>
```

当 HTML 文件嵌入 JavaScript 程序代码后，浏览器程序在读到＜script＞标记时，就解释执行其中的脚本。其中，＜script＞标记可以放在 Web 页面的＜head＞…＜/head＞标记中，也可以放在＜body＞…＜/body＞标记中。例如，在＜body＞…＜/body＞标记中可以输入以下代码：

```
<script type="text/javascript">
    alert("我要去学习 JavaScript!")          //弹出警告框
</script>
```

需要注意的是，JavaScript 脚本可以放在＜body＞标记中的任何位置。如果所编写的 JavaScript 程序用于输出网页的内容，应该将 JavaScript 程序置于 HTML 文件中需要显示该内容的位置。

5.2.2 在 HTML 页面中链接外部的 JavaScript 文件

在 Web 页面引入 JavaScript 的另一种方法是采用链接外部 JavaScript 文件的形式。如果脚本代码比较复杂或是同一段代码需要被多个页面使用，则可以将这些脚本代码放置在一个单独的文件中(保存文件的扩展名是.js)，然后在需要使用该代码的 Web 页面中链接该 JavaScript 文件即可。

在 Web 页面中链接外部 JavaScript 文件的语法格式如下：

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

需要注意的是，调用外部文件 myjs.js 时，首先需要编写外部的 JavaScript 文件，并命名为 myjs.js。然后，在 HTML 页面中调用外部的 JavaScript 文件 myjs.js。

5.3 JavaScript 语法

5.3.1 JavaScript 的基本语法规则

每一种计算机语言都有自己的基本语法，学好基本语法是学好编程语言的关键。同样，学习 JavaScript 语言，也需要遵从一定的语法规范，如执行顺序、大小写问题以及注释

语句等。

1. 执行顺序

JavaScript 程序按照在 HTML 文件中出现的顺序逐行执行。如果某些代码(例如函数、全局变量等)需要在整个 HTML 文件中使用,最好将其放在 HTML 文件的<head>...</head>标记中。某些代码,如函数体内的代码,不会被立即执行,只有当所在的函数被其他程序调用时,该代码才会被执行。

2. 区分大小写

JavaScript 严格区分字母大小写。也就是说,在输入关键字、函数名、变量及其他标识符时,都必须采用正确的大小写形式。例如,变量 username 与变量 userName 是两个不同的变量。

3. 每行结尾的分号可有可无

JavaScript 语言并不要求必须以分号(;)作为语句的结束标记。如果语句的结束处没有分号,JavaScript 会自动将该行代码的结尾作为语句的结尾。例如,下面两行代码都是正确的。

```
alert("您好,欢迎学习 JavaScript!")
alert("您好,欢迎学习 JavaScript!");
```

应该注意的是,最好的代码编写习惯是在每行代码的结尾处加上分号,这样可以保证代码的严谨性和准确性。

4. 注释

在编写程序时,为了使代码易于阅读,通常需要为代码加一些注释。注释是对程序中某个功能或者某行代码的解释、说明,用来提高代码的可读性,而不会被 JavaScript 当成代码执行。

JavaScript 为开发人员提供了两种注释形式:单行注释和多行注释,具体如下:

(1) 单行注释使用双斜线"//"作为注释标记,将"//"放在一行代码的末尾或者单独一行的开头,它后面的内容就是注释部分。

(2) 多行注释可以包含任意行数的注释文本。多行注释是以"/*"标记开始,以"*/"标记结束,中间的所有内容都为注释文本。这种注释可以跨行书写,但不能有嵌套的注释。

下面是合法的 JavaScript 注释:

```
//这里的单行注释
/*这里是一段注释*/  //这里是另一段注释
/*这里是多行注释
*/
```

5.3.2 变量的声明与赋值

在 JavaScript 中,使用变量前需要先对其进行声明。所有的 JavaScript 变量都由关

键字 var 声明，语法格式如下：

```
var  abc;
```

在声明变量的同时也可以对变量进行赋值，例如：

```
var  abc=1;
```

声明变量时，需要遵循的规则如下：

(1) 可以使用一个关键字 var 同时声明多个变量，例如：

```
var a,b,c                          //同时声明 a、b 和 c 三个变量
```

(2) 可以在声明变量的同时对其赋值，即初始化，例如：

```
var a=1,b=2,c=3;                   //同时声明 a、b 和 c 三个变量，并分别对其进行初始化
```

(3) 如果只是声明了变量，并未对其赋值，则其默认为 undefined。

(4) var 语句可以用作 for 循环和 for/in 循环的一部分，这样就使循环变量的声明成为循环语法自身的一部分，使用起来比较方便。

(5) 使用 var 语句多次声明同一个变量，如果重复声明的变量已经有一个初始值，那么此时的声明就相当于对变量的重新赋值。

当给一个尚未声明的变量赋值时，JavaScript 会自动用该变量名创建一个全局变量。在一个函数内部，通常创建的只是一个仅在函数内部起作用的局部变量，而不是一个全局变量。创建一个局部变量，并不是简单地赋值给一个已经存在的局部变量，必须使用 var 语句进行变量的声明。

另外，由于 JavaScript 采用弱类型的形式，可以不理会变量的数据类型，即可把任意类型的数据赋值给变量。

例如，声明一些变量，具体代码如下：

```
var a=100                          //数值类型
var str="网页平面设计学院"          //字符串类型
var bue=true                       //布尔类型
```

值得注意的是，在 JavaScript 中，变量可以先不声明，而在使用时，根据变量的实际作用确定其所属的数据类型，为了良好的编程习惯和能够及时发现代码中的错误，建议在使用变量前对其声明。

5.3.3 函数

在 JavaScript 中，经常会遇到程序需要多次重复操作的情况，这时就需要重复书写相同的代码，这样不仅加重了开发人员的工作量，而且对于代码的后期维护相当困难。为此，JavaScript 提供了函数，它可以将程序中烦琐的代码模块化，提高程序的可读性，并且便于后期维护。

1. 函数定义

为了使代码更为简洁并可以重复使用，通常会将某段实现特定功能的代码定义成一

个函数。在 JavaScript 程序设计中，所谓函数就是在计算机程序中由多条语句组成的逻辑单元，在 JavaScript 中，函数使用关键字 function 定义，其语法格式如下：

```
<script type="text/javascript">
    function 函数名 ([参数 1,参数 2,...]){
        函数体
}
</script>
```

从上述语法格式可以看出，函数的定义由关键字 function、“函数名”、“参数”和“函数体”4 部分组成，关于这 4 部分的相关说明如下：

- function——在声明函数时必须使用的关键字。
- 函数名——创建函数的名称，是唯一的。
- 参数——外界传递给函数的值，是可选的，当有多个参数时，各参数用“,”分隔。
- 函数体——函数定义的主体，专门用于实现特定的功能。

对函数定义的语法格式有所了解后，下面定义一个无参函数 show()，并在函数体中输出“欢迎光临，网页平面设计学院”，具体示例如下：

```
<script type="text/javascript">
    function show(){
        alert("欢迎光临,网页平面设计学院");
}
</script>
```

上述代码定义的 show()函数较简单，它没有定义参数，并且函数体中仅使用 alert()语句返回一个字符串。

2. 函数的调用

当函数定义完成后，要想在程序中发挥函数的作用，必须调用这个函数。函数的调用非常简单，只需引用函数名，并传入相应的参数即可。函数调用的语法格式如下：

```
函数名称([参数 1,参数 2,...])
```

在上述语法格式中，“[参数 1,参数 2,...]”是可选的，用于表示参数列表，其值可以是一个或多个。

为了使初学者能够更好地理解函数调用，下面通过一个案例演示函数的调用。

例 5-1 函数调用示例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
</head>
```

```
<body>
<button onclick="show()">点击这里</button>  <!--通过鼠标点击事件调用函数-->
</body>
</html>
<script type="text/javascript">
    function show(){
        alert("欢迎光临");
    }
</script>
</body>
</html>
```

在上述代码中，首先定义了一个名为 show()的函数，该函数比较简单，仅使用 alert()语句返回一个字符串，然后在按钮 onclick 事件中调用 show()函数。其中本案例使用的 onclick 事件将在后面做具体介绍，此处了解即可。

运行例 5-1，结果如图 5-3 所示。单击图 5-3 中的按钮，即会弹出图 5-4 所示的警示框。

图 5-3 函数调用 1

图 5-4 函数调用 2

5.3.4 JavaScript 中的对象

1. 对象简介

JavaScript 所实现的动态功能，基本上都是对 HTML 文档或者 HTML 文档运行环境进行的操作。那么要实现这些动态功能就必须找到相应的对象。JavaScript 中有已经定义过的对象供开发者调用，在了解这些对象之前先看图 5-5 所示的内容。

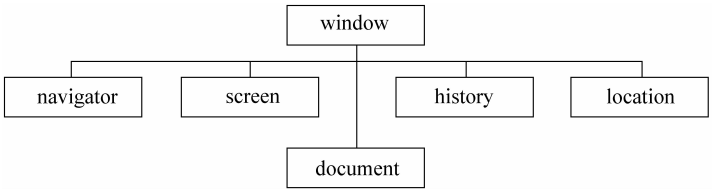

图 5-5 在浏览器窗口中的文档对象模型

图 5-5 中的内容是一个简单的 HTML 文档在浏览器窗口中的文档对象模型，其中

window、navigator、screen、history、location 都是 HTML 文档运行所需的环境对象，document 对象才是前面讲述的 HTML 文档，当然这个 document 对象还可以划分出 html、head、body 等分支。

- window 对象是所有对象中最顶层的对象，HTML 文档在 window 对象中显示。
- navigator 对象可以读取浏览器相关的信息。
- screen 对象可以读取浏览器运行的物理环境，例如屏幕的宽和高，单位为像素。
- document 对象是整个网页 HTML 内容，每个 HTML 文档被浏览器加载以后都会在内存中初始化一个 document 对象。
- history 对象可以控制浏览器的前进和后退。
- location 对象可以控制页面的跳转。

这些对象中，较常用的有 window 对象、document 对象和 location 对象。

2. window 对象

window 对象是所有 JavaScript 对象中最顶层的对象，整个 HTML 文档就是一个浏览器窗口，当打开一个浏览器窗口以后，不管有没有内容，都会在内存中形成一个 window 对象。window 对象所提供的方法很多，在下面的内容中介绍最常用的几种方法。

1）窗体的创建和关闭

利用 window 对象可以新建浏览器窗口，也可以关闭浏览器窗口，下面来看具体的操作代码。

例 5-2 窗体的创建和关闭示例。

```
<html>
<head>
<title>窗体的创建和关闭示例</title>
<script type="text/javascript">
    var win;
    function createWin() {
        win=window.open("","","width=300,height=200");
    }
    function closeWin() {
        if (win) {
            win.close();
        }
    }
</script>
</head>
<body>
<form>
<input type="button" value="创建新窗口" onclick="createWin()">
<input type="button" value="关闭新窗口" onclick="closeWin()">
</form>
```

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

这个程序在浏览器中运行以后，界面上会有两个按钮，单击“创建新窗口”按钮会弹出一个新的浏览器窗口，这个窗口的宽为 300 像素、高为 200 像素；单击“关闭新窗口”按钮，这个弹出窗口就会被关闭。

上面这个程序中用到的就是 window 对象的 open 和 close 两个方法，open 方法新建一个窗口，close 方法关闭指定窗口。

2）三种常用的对话框

在 window 对象中，有三种常用的对话框：第一种是警告对话框，第二种是确认对话框，第三种是输入对话框。下面这个示例中展示了这三个对话框的用法。

例 5-3 三种常用的对话框。

```
<html>
<head>
<title>三种常用的对话框</title>
<script type="text/javascript">
    function alertDialog() {
    alert("您成功执行了这个操作。");
    }
    function confirmDialog() {
        if(window.confirm("您确认要进行这个操作吗？"))
            alert("您选择了确定！");
        else
            alert("您选择了取消");
    }
    function promptDialog() {
        var input=window.prompt("请输入内容:");
        if(input !=null)
        {
            window.alert("您输入的内容为"+input);
        }
    }  </script>
</head>
<body>
<form>
<input type="button" value="警告对话框" onclick="alertDialog()">
<input type="button" value="确认对话框" onclick="confirmDialog()">
<input type="button" value="输入对话框" onclick="promptDialog()">
</form>
</body>
</html>
```