第5章 JavaScript基础

JavaScript 是一种解释型的脚本语言，它可以嵌入 HTML 页面中，并在客户端执行，是目前绝大多数浏览器普遍支持的脚本语言。JavaScript 是动态 Web 设计的最佳选择，被广泛应用于 Web 应用开发，来为网页添加各式各样的动态功能，为用户提供更流畅美观的浏览体验。JavaScript 的解释器被称为 JavaScript 引擎，是浏览器的一个重要组成部分。

5.1 JavaScript 简介

JavaScript 是目前非常流行的一种 Web 前端的描述性脚本语言。它是基于对象(Object)的、事件驱动的、并具有相对安全性的客户端脚本语言。JavaScript 运行在客户端，从而可以减轻服务器端的负担。

HTML 网页在互动性方面能力较弱。JavaScript 是一种嵌入 HTML 中的描述性语言，它并不编译产生机器代码，只是由浏览器的解释器将其动态地处理成可执行的代码。JavaScript 起源于 LiveScript 语言。当初，使用 HTML 表单与用户的交互，成了制约网络发展的重大瓶颈。于是 Netscape 公司推出了 LiveScript 语言，并最终定名为 JavaScript。后来，有 3 种不同版本的 JavaScript 版本，即 Netscape 的 JavaScript、微软的 Jscript 以及 CEnvi 的 ScriptEase。最终，ECMA 于 1997 年确定了 JavaScript 的标准，并被 ISO 采纳通过，作为浏览器使用的脚本语言的统一标准。如今，JavaScript 已经成为 Web 浏览器中不可缺少的一种技术。随着 HTML5 的推广与广泛应用，大量基于 JavaScript 的跨平台框架和游戏引擎产生，大大地扩展了 JavaScript 的应用范围。

JavaScript 脚本语言具有如下一些特点。

(1) 它是一种脚本编程语言，采用小程序段的方式实现编程，是一种解释性语言，无须编译，在代码运行过程中被逐行地解释。它提供了一个简易的开发过程，与 HTML 结合在一起，方便用户使用操作。它的数据是弱类型的，未使用严格的类型检查。

(2) 它是一种基于对象的语言,许多功能可以来自脚本环境中对象的方法与脚本的相互作用。

(3) 它具有简单性,它的基本语句和控制流设计简单而紧凑,它的变量类型采用弱类型。

(4) 它是一种安全性语言,不允许任何人访问本地硬盘,不能将数据存放到服务器上,不允许对网络文档进行删除和修改,只能通过浏览器实现信息浏览与互动,从而有效地防止数据丢失,保障数据的安全性。

(5) 它是动态的,可以直接对用户的输入做出响应,无须经过 Web 服务程序。它对用户的反应采用事件驱动的方式。

(6) 它具有跨平台性。它依赖浏览器本身,与操作环境和机器硬件系统无关。

JavaScript 脚本语言有如下一些典型应用。

1) 将 JavaScript 脚本嵌入 HTML 页面中,实现动态文本、动态窗口及动画效果。

2) 对浏览器事件做出响应。

3) 读写 HTML 元素信息,在数据被提交到服务器之前进行验证。

4) 检测用户的浏览器信息,创建与编辑 cookies 信息。

5) 基于 Node.js 技术的服务器端编程。

5.2 JavaScript 脚本的使用

5.2.1 JavaScript 脚本的应用实例

JavaScript 脚本在 HTML 页面中必须放置在<script>与</script>标签之间,这样浏览器才能解释和运行这对标签中的代码。如果没有这对标签,直接把 JavaScript 脚本放置在页面中,浏览器会把脚本的内容当成纯文本来处理,就不会实现脚本真正的运行效果。基本语法如下。

```
<script type="text/javascript"  [src="外部 JavaScript 文件"]>
      …
</script>
```

其中,type 属性定义脚本的 MIME 类型。MIME 类型由媒介类型和子类型两部分组成,用"/"分隔。JavaScript 脚本的 MIME 类型为"text/javascript"。几乎所有的现代浏览器都把 JavaScript 作为默认的脚本语言,所以,好多情况下这个 type 属性可以省掉。src 属性定义要加载的外部 JavaScript 脚本文件的位置。如果不使用外部的 JavaScript 脚本文件,则可以省略这个属性。

【例 5-1】 JavaScript 脚本的应用示例。运行效果如图 5-1 所示。

```
<html>
  <head>
    <title>JavaScript 脚本应用示例</title>
  </head>
```

```
  <body>
    <script type="text/javascript">
    document.write("这是第一个 JavaScript 脚本实例");
    </script>
  </body>
</html>
```

运行这个实例后，页面上会显示“这是第一个 JavaScript 脚本实例”这样一行文本。

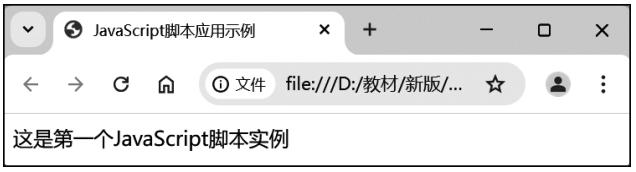

图 5-1　JavaScript 脚本的应用示例

5.2.2　JavaScript 脚本的引用方法

JavaScript 脚本不能独立运行，必须依附于某个网页，在浏览器端运行。按照 JavaScript 脚本与它所依附的 HTML 页面之间的关系，一般分为三种引用方式。一种方式是直接将 JavaScript 脚本嵌入 HTML 文档中，成为 HTML 文档的一部分。另一种方式是将 JavaScript 脚本单独放置在一个 JavaScript 文件中，再引入 HTML 文档。第三种方式是将 JavaScript 脚本作为某些特定标签的属性值来使用。具体使用哪种方式，要依据情况而定。

1. 在 HTML 文档中嵌入 JavaScript 脚本

在 HTML 文档中，可以将脚本语句放置在＜script＞＜/script＞标签对之间。每个标签对中可以包含多段脚本语句块。各个＜script＞＜/script＞标签对中的脚本块之间可以相互访问。基本语法如下。

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

其中，type 属性定义脚本的 MIME 类型。MIME 类型由媒介类型和子类型两部分组成，用“/”分隔。JavaScript 脚本的 MIME 类型为“text/javascript”。

包含有 JavaScript 脚本的＜script＞＜/script＞标签可以放置在 HTML 文档的 head 部分或 body 部分中。如果把＜script＞标签放置在 head 部分，页面载入的时候同时载入了脚本代码，然后可以在 body 部分调用。通常可以在 head 部分放置准备为 body 部分调用的各类函数或全局变量的声明等内容。其基本使用形式如例 5-2 所示。

【例 5-2】　JavaScript 脚本放置在页面文档的 head 部分示例。运行效果如图 5-2 所示。

```
<html>
  <head>
    <title>JavaScript 脚本放置在页面文档的 head 部分示例</title>
    <script  type="text/javascript">
      document.write("JavaScript 脚本放置在页面文档的 head 部分的效果。");
    </script>
```

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

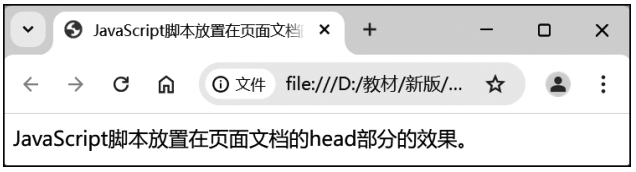

图 5-2　JavaScript 脚本放置在页面文档的 head 部分示例

另一种嵌入方式，是将脚本的<script>标签放置在 body 部分中，在页面载入的时候需要同时执行，这些代码执行后的输出成为页面的内容，在浏览器中可以即时看到。一般来说，要在页面加载过程中运行，动态建立一些 Web 页面的内容时，这些脚本应放在 body 中。而把脚本代码定义为函数时，用于处理页面事件的脚本代码应放在 head 中，这样可以使其在 body 之前加载。这种使用形式如例 5-3 所示。

【例 5-3】 JavaScript 脚本放置在页面文档的 body 部分示例，运行效果如图 5-3 所示。

```
<html>
  <head>
    <title>JavaScript 脚本放置在页面文档的 body 部分示例</title>
  </head>
  <body>
    <script  type="text/javascript">
      document.write("JavaScript 脚本放置在页面文档的 body 部分的效果。");
    </script>
  </body>
</html>
```

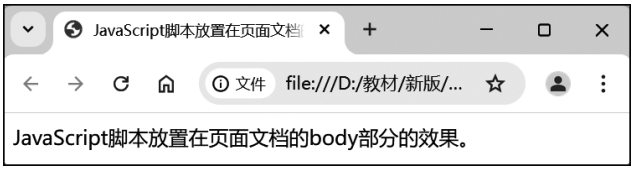

图 5-3　JavaScript 脚本放置在页面文档的 body 部分示例

2. 将外部的 JavaScript 脚本文件引入 HTML 文档中

如果把大量的 JavaScript 脚本直接写进 HTML 文档中，会使 HTML 文档比较臃肿，既不方便阅读，也不利于维护。而且好多时候，一些脚本内容需要被多个页面共享。如果把脚本代码组织成一个单独的文本文件，当 HTML 文档需要使用这些脚本时，再通过引用的方式引入 HTML 文档中，就可解决以上提到的问题。JavaScript 文件是一种文本文件，它的文件扩展名是“.js”。在 HTML 文档中引用外部 JavaScript 文件时，其基本语法格式如下。

```
<script type="text/javascript" src="JavaScript 文件的路径"></script>
```

其中，type 属性如前所述，定义脚本的 MIME 类型。src 属性定义所引用的外部 JavaScript 文件的路径。

【例 5-4】 引用外部 JavaScript 脚本示例。运行效果如图 5-4 所示。

```
<html>
  <head>
    <title>引用外部 JavaScript 脚本示例</title>
    <script  type="text/javascript" src="example.js"></script>
  </head>
  <body>
  </body>
</html>
```

其中，example.js 文件中的内容为：document.write("引用外部 JavaScript 脚本的效果。")。

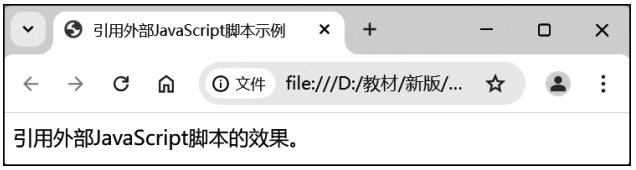

图 5-4 引用外部 JavaScript 脚本示例

3. 将 JavaScript 脚本作为特定标签的属性值来使用

可以通过“javascript:XXX”的形式来调用 JavaScript 的函数或方法，并将它作为某个标签的属性值来使用，其中的“XXX”表示具体的脚本内容，如例 5-5 所示。

【例 5-5】 JavaScript 脚本作为标签属性值示例。

```
<html>
  <head>
    <title>JavaScript 脚本作为标签属性值示例</title>
  </head>
  <body>
    <a  href="javascript:alert('JavaScript 脚本应用举例。')"></a>
  </body>
</html>
```

或者也可以使 JavaScript 脚本和用户事件结合起来，用脚本的运行作为事件的响应。如例 5-6 所示。

【例 5-6】 JavaScript 脚本与标签的事件结合使用示例。

```
<html>
  <head>
    <title>JavaScript 脚本与标签的事件结合使用示例</title>
  </head>
  <body>
    <span style="cursor:hand" onclick="alert('JavaScript 脚本与标签的事件结合使
用举例。')"></span>
  </body>
</html>
```

在上例中，onclick 属性表示对标签单击事件的响应，alert()函数是对应的响应脚本。

5.3 JavaScript 的语法与数据类型

5.3.1 基本语法

1. 语句和语句块

语句是构成 JavaScript 脚本的基本单位。JavaScript 语句通常由字面量、变量、运算符、表达式和关键字等组成。JavaScript 以分号作为语句结束的标志，但这并不是强制性的要求，如果语句的结束处没有分号，会自动以该行代码的结尾作为语句的结束。所以，如果有多条语句写在同一行中，应当在语句的结束处加上分号。JavaScript 解释器的语法检查不太严格，程序员编写 JavaScript 代码时最好能用比较严谨的书写风格，这样才不会导致程序语句的二义性，也使代码更清晰，便于阅读。

在逻辑上相关的若干条语句可以放置在“{ }”内，称为一个语句块，可以形成相对完整的逻辑功能，有助于更清晰、准确地定义逻辑边界。语句块通常会应用于条件语句、循环语句及函数当中。

2. 空白符

JavaScript 会忽略程序代码中字符串以外的空白符(空白符包括空格符、换行符和制表符等)。编写代码时，可以灵活运用空白符来排版，提高代码的可读性。

3. 标识符的大小写

JavaScript 语句对大小写是敏感的，比如变量 area 和变量 Area 表示的是两个不同的变量。所以，在代码的编写过程中，要对关键字、变量、函数名及其他标识符中的大小写进行严格区分，保证正确的书写形式。

4. 注释

为了增强代码的可读性，便于对代码进行修改和维护，可以在程序中使用注释。解释器在解释程序时，会忽略注释部分。注释有单行注释和多行注释两种形式。

单行注释用两个斜杠“//”来表示，从代码中的“//”处开始直到本行结束处，都是注释的内容。多行注释则从代码中的“/*”处开始，到“*/”处结束。

注释还有另外一个作用，就是用来屏蔽某些语句，使程序能够暂时忽略这些语句。这种用法通常用在调试代码时。

5.3.2 数据类型

数据类型是编程中所使用的一组性质相同的值的集合及定义在其上的操作的总称。JavaScript 是一种弱类型语言。编程时可以不提前声明变量的类型，在变量被赋值或使用时，其类型会被自动确定。当然，也可以先声明变量的数据类型，再进行赋值和使用。JavaScript 中的数据类型可以分为三个类别，分别是基本数据类型、复合数据类型和特殊数据类型，下面分别介绍。

1. 基本数据类型

JavaScript 的基本数据类型包括数值型(Number)、字符串型(String)和布尔型(Boolean)三种类型。

1) 数值型(Number)

数值型既包括整型数也包括浮点型数,如 12、25.8、036、0x9A、3.58e11 等。在这几种表示法中,以"0"打头的 036 表示八进制数 36,以"0x"打头的 0x9A 表示十六进制数 9A,3.58e11 是使用科学计数法表示的数值,它的值是 3.58×10^{11}。除此以外,数值型还包括三个特殊的值:NaN、Infinity、-Infinity,分别表示"非数值"、"正无穷大"和"负无穷大"的含义。

2) 字符串型(String)

字符串型是有限个 Unicode 字符组成的序列,用于表示文本数据。字符串型数据书写时用英文的单引号或双引号括起来。单引号定界的字符串中可以含有双引号,同样,双引号定界的字符串中可以含有单引号,但是字符串中不能再包含同样定界符的字符串。以下是字符串书写方法的例子。

```
"Nowadays almost all web pages contain JavaScript" //合法,使用双引号定界
'Nowadays almost all web pages contain JavaScript' //合法,使用单引号定界
"Nowadays almost all web pages contain 'JavaScript'" //合法,双引号定界串内使用单引号
'Nowadays almost all web pages contain "JavaScript"' //合法,单引号定界串内使用双引号
"Nowadays almost all web pages contain "JavaScript"" //非法,双引号定界串内使用双引号
```

字符串中包含的字符个数称为字符串的长度,如字符串"JavaScript"的长度是 10,字符串"Hello World!"的长度是 12。空字符串也叫空串,是指不包括任何字符的字符串,即" "。空字符串的长度是 0。

如果字符串中引号的使用与定界符造成冲突时,可以使用转义字符来表示串内的引号。一些不可显示的特殊字符也可以用转义字符来表示。转义字符是由"\"开头,后跟一个或几个字符组成的。转义字符具有特定的含义,不同于字符原有的含义,所以称作"转义字符"。表 5-1 是 JavaScript 中常用的转义字符。

表 5-1 JavaScript 中常用的转义字符

转义字符	描述	转义字符	描述
\b	退格	\\	反斜杠
\n	换行	\r	回车符
\t	水平制表符	\ooo	用八进制 ASCII 码表示的字符
\'	单引号	\xHH	用十六进制 ASCII 码表示的字符
\"	双引号	\uhhhh	用十六进制编码表示的 Unicode 字符

3) 布尔型(Boolean)

布尔型数据只有两个取值,分别是 true(真)和 false(假)。布尔型数据是用来进行逻辑判断的,说明了某个命题或判断是真的还是假的。比如 5>3 的结果是 true,而 0==1 则是

假的。每个关系表达式的运算结果都是一个布尔型的值。

布尔型数据通常用于 JavaScript 代码中的控制结构。在 if…else…语句中，可以通过判断条件表达式值的结果来控制程序流程。如果条件表达式的值为 true 时执行一种动作，条件表达式的值为 false 时执行另一种动作。

2. 复合数据类型

JavaScript 的复合数据类型是由一些基本的数据类型复合而成的。复合数据类型也称为引用数据类型。JavaScript 中的复合数据类型包括数组型(Array)和对象型(Object)。

1) 数组型(Array)

在 JavaScript 中，数组用来存放一组相同或不同类型的数据。JavaScript 中的数组是一个功能十分强大的“容器”，它不仅可以代表数组，也可作为长度可变的线性表来使用。数组中存放的数据称为数组的“元素”，数组的元素个数是可变的。未赋值的数组元素的值为 undefined。

2) 对象型(Object)

对象是对现实世界中的事物的抽象。JavaScript 的对象中保存的是一组描述对象特征的属性和反映对象行为的方法(也称为函数)。同一种对象类型具有相同的属性类别和方法。通过对象名可以访问对象的属性和方法。JavaScript 还提供了大量的内置对象，供用户使用。

3. 特殊数据类型

JavaScript 的特殊数据类型是指无法归入前面两种数据类型中的特殊数据。特殊数据类型包括 null 和 undefined。

1) 空值(null)

JavaScript 中的关键字 null 是一个特殊的值，用来表示空的或不存在的引用。如果试图去引用一个没有定义的变量，会返回 null 值。需要注意的是，null 值不等同于空字符串或 0，同时也不等同于 undefined。null 不可以写成 Null 或 NULL。可以通过将变量的值设为 null 来清空变量。

2) 未定义类型(undefined)

未定义类型即 undefined，一个变量创建后还没有赋值，或者赋予了一个不存在的属性值时，该变量的值就是 undefined。当引用一个不存在的数组元素或对象属性时，会返回 undefined。

虽然 null 和 undefined 都具有“空值”的含义，但它们之间还具有完全不同的含义。null 表示一个变量被赋予了一个空值，而 undefined 表示变量尚未被赋值，不含有明确的值。可以通过将变量的值设为 null 来清空变量。

如果定义的变量准备将来保存对象，那么最好将该变量初始化为 null。这样，只要直接检测 null 值，就可以知道相应的变量是否已经保存了一个对象的引用。

5.3.3 常量与变量

1. 常量

程序运行过程中值保持不变的数据称为常量。常量可以为程序提供固定的和精确的数

值。根据数据的表现形式可以确定常量的类型，如 425 是数值型常量，“take”是字符串型常量，true 是布尔型常量等。在 JavaScript 脚本中，可以直接输入和使用这些值。常量在程序中被定义后，便会在计算机内存中的固定位置存储下来，在程序结束之前，它是不会发生变化的。

2. 变量

变量是指程序中一个命名的存储单元，是存取数字、提供存放信息的容器。存储在这个存储单元中的值在程序运行过程中可以随意地改变。变量是通过变量名来标识的。

1）变量名

变量名用来标识程序中的变量，在同一段程序中，变量名是唯一的。JavaScript 中的变量名是区分大小写的。为了增强程序的可读性，变量命名时应具有一定的含义。JavaScript 中变量的命名规则如下。

变量名中只能包含字母、数字和下画线三种字符，且只能以字母或下画线开头；变量名不能使用 JavaScript 中的关键字；变量名的长度原则上没有限制。

2）变量的声明

在 JavaScript 中，变量在使用前需要先声明，所有类型的变量都用关键字 var 来声明。声明变量的语法形式如下。

```
var  variablename;
```

其中的 variablename 是要声明的变量名。如果要同时声明多个变量，变量名之间用逗号分隔。如：

```
var  variablename1, variablename2, variablename3;
```

或者也可以不使用 var 关键字，而是直接通过赋值方式定义变量。如：

```
param="hello";
```

3）变量的赋值

声明变量的同时，可以使用赋值号对变量进行初始化赋值。其语法形式如下。

```
var  area =300;
```

其中赋值号左边的 area 是要进行初始赋值的变量，赋值号右边是所赋的值。另外，还有一种用法是先声明变量，之后再对变量进行赋值。其语法形式如下。

```
var  area;
area =300;
```

由于 JavaScript 是一种弱类型语言，所以变量可以无须声明而直接进行赋值。但是，建议最好还是在使用变量之前先声明，这样便于发现代码中的错误，有利于编程的排错和调试。

4）变量的类型

变量的类型是指变量的值所属的数据类型。JavaScript 是一种弱类型语言，所以可以将任意类型的数据赋值给变量。而且在程序运行过程中，可以给同一个变量赋不同类型的

值。如：

```
var  area =5 * 20;
area ="长方形的面积是" +100;
```

5）变量的作用域和生存期

变量的作用域是指变量在程序中发挥作用的有效范围，也就是这个变量可以被使用的区域。在 JavaScript 中，变量按作用域可以分为两种，即全局变量和局部变量。全局变量定义在所有函数之外，在整个脚本代码中都可以使用。局部变量是定义在某个函数的函数体内的变量，只能在定义它的函数体范围内使用，在其他函数内不可见。

对于脚本中需要使用的全局变量，一般可以在页面的＜head＞部分声明，在页面的＜body＞部分使用。

变量的生存期是指变量在程序运行过程中，在计算机内的有效存在期。全局变量的生存期从它被定义的时刻开始，直到主程序结束为止。局部变量的生存期从定义它的函数被执行的时刻开始，函数执行结束后，局部变量的生存期也随之结束。所以，通常来讲，全局变量比局部变量的生存期要长。

5.3.4 表达式与运算符

1. 表达式

表达式是变量、常量以及运算符的组合，可以完成赋值、计算等一系列操作。表达式可分为算术表达式、字符串表达式、赋值表达式和布尔表达式等。

2. 运算符

运算符是表示数据处理方式的一种符号，用于对一个或多个运算对象进行运算，以实现自然算法的计算机表示，是完成一系列操作的基础。在 JavaScript 中，按功能划分，运算符可以分为算术运算符、比较运算符、赋值运算符、字符串运算符、逻辑运算符、条件运算符等。

1）算术运算符

算术运算符用于进行基本的算术运算，包括加、减、乘、除、取模、自增和自减等运算。运算的结果是数值。JavaScript 中常用的算术运算符如表 5-2 所示。

表 5-2 JavaScript 中常用的算术运算符

运 算 符	描 述	示 例	功 能
＋	加法运算	5＋3	计算两个操作数的和
－	减法运算	100－58	计算两个操作数的差
*	乘法运算	2 * 3	计算两个操作数的积
/	除法运算	15/5	计算两个操作数的商
%	取模运算	8%3	计算前一个操作数除以后一个操作数的余数
＋＋	自增运算	a＋＋ ＋＋a	变量的值增 1，返回变量的原值 变量的值增 1，返回变量的新值
－－	自减运算	a－－ －－a	变量的值减 1，返回变量的原值 变量的值减 1，返回变量的新值