JavaScript

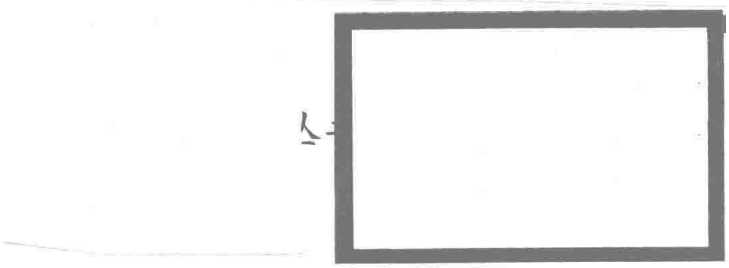
页面效果大师

怀石工作室
白鉴聪　王进　编著

从众多实例中融会贯通，制作个性化的网页效果！

网络开发实例精解系列丛书

机械工业出版社
China Machine Press



JavaScript 网页效果大师

前 言

JavaScript 是当前网站效果设计的最成熟的开发语言，也是最广泛应用的网页脚本语言之一。由于 JavaScript 是专门用于 Web 页的交互性设计的，因此特别适合网站开发人员设计各种页面效果。它的强大功能加上开发人员的巧妙构思，几乎无所不能，无论是绚丽夺目还是生动活泼的效果都可以使用 JavaScript 来实现。为此，本书专门结合大量的效果实例，介绍如何使用 JavaScript 来制作各种网页效果。

本书主要分为三大部分。第一部分为本书的第 1 篇“基础篇”，它对 JavaScript 的历史和特性进行了简要介绍，并且详细说明了 JavaScript 的语法与常用对象。第二部分为本书的主体，它根据 JavaScript 脚本实现的效果的侧重点，分为“窗口篇”、“按钮篇”、“文字篇”、“图像篇”、“背景和链接篇”、“时间篇”、“页面篇”、“鼠标篇”、“系统篇”、“综合篇”介绍了 99 个实例。第三部分为本书的最后一篇“参考手册篇”，它按照字母顺序详细介绍了 JavaScript 中常用的对象，以及各个对象的属性、方法和事件处理函数。在学习各个实例的过程中，或者在用 JavaScript 进行创作的过程，如果你对要用到的 JavaScript 对象、属性、方法和事件处理函数有不明白的地方，可以查阅这个参考手册。

本书所介绍的效果实例已经囊括了当前网站的大多数网页效果，因此在阅读完本书后，或许你已经不会再对所看到的网页效果感到惊奇，因为你也可以做到，甚至可以比它们做得更好。为了使大家能够更好地理解每个网页效果实例，而且希望大家有进一步的提高，本书的实例采用了**理论与程序相结合**的结构：**实现原理+技术分析+实现代码+总结**。实现原理部分，清楚详细地介绍了每个实例实现的原理思想，对于一些比较复杂的实例，还提供了清晰的实例流程图。技术分析部分，详细地介绍了当前实例需要用到的 JavaScript 的技术、对象等，还介绍了各种 JavaScript 对象在实际程序设计中可能应用到的方面——这是我们的编写人员多年使用 JavaScript 开发的宝贵经验。实现代码部分，不仅重点解释一些关键语句的意思，而且在代码后面还介绍了整个程序的实现过程和程序内部各个函数的相互关系，使大家不必为程序代码的理解而烦恼。总结部分，不仅演示了整个实例效果，而且还提出了可以改进提高的地方和个人制作的经验体会。

编写本书的目的不仅是使大家明白特定的网页效果的制作过程，更重要的是希望大家能够理解并掌握一个有普遍意义的效果实例设计前后的思想和原理，从而使自身程序设计水平得到提高。大家在阅读本书时，不妨多思考每个实例的制作思想，多体会实际代码的含义。

本书面向的是掌握基本 HTML 语言、有上网经验的读者，而已经学习过 JavaScript 的读者可以更好地学习本书中实例，重点体会每个实例的设计思想和实现技术。如果对 JavaScript 只是有初步了解甚至没有学过的读者，可以先学习本书的基础篇，然后结合每个实例的技术分析部分的内容再深入学习 JavaScript。对于一些已经能够熟练使用 JavaScript 开发的读者，可以将本书作为参考手册，以备需要时能够快速开发。

本书共分基础篇、窗口篇、按钮篇、文字篇、图像篇、背景+链接篇、时间篇、页面篇、鼠标篇、系统篇、综合篇和参考手册篇。读者可以根据自己需要设计的效果类型到相应篇中进行学习。其中窗口篇介绍的是对窗口的控制，包括窗口的大小、开启和关闭等；按钮篇介绍的各种按钮效果以及一些应用在按钮上的功能的实现；文字篇介绍的文字效果覆盖了当前网站使用了的大多数效果；图像篇介绍了如何对图片进行控制和图片效果方面的处理机制；背景+链接篇主要介绍了在这两方面的效果设计；时间篇介绍了时间方面效果的设计技巧，包括显示当前日期、制作月历等；页面篇重点介绍了对网页的控制技巧，包括将页面添加到收藏夹，自动设置首页等；鼠标篇主要介绍了对鼠标的控制技巧以及利用鼠标设计的一些小花样；系统篇介绍了如何利用用户本地系统为网页服务，包括获取用户系统信息、打印当前网页等；综合篇则介绍了多个网上常用的工具，包括搜索引擎、计算器等。

本书的实例内容十分丰富，而且每个实例都有详细的说明，读者可以根据自身需要选择不同篇章学习，不一定要按顺序阅读，因为每篇所需要用到的知识和技术都已经在该篇内专门介绍。

希望大家在学习了本书后，即使不能成为 JavaScript 网页效果大师，至少也可以成为一名 JavaScript 效果设计高手。如果你有什么疑问、困难或者对本书的意见，那么可以通过邮件和我们联系，我们真诚地希望与大家进行交流。

Email：ebook@mail.machineinfo.gov.cn，huaishi@elong.com。

怀石工作室

2001 年 8 月

目 录

第 1 篇 基础篇

JavaScript 是一种开发网页所使用的脚本语言。它能够使网页具有更好的交互性，还能够给网页添加生动活泼的内容。可以说，使用了 JavaScript 的网页，具有一股灵气。如果你的网页中没有一句 JavaScript 代码，那么这些网页可就显得“太业余”了，也不会非常吸引人。下面我们就来看一下究竟什么是 JavaScript。

JavaScript 简介

JavaScript 是由 Netscape 公司随 Navigator 一起发布的一种脚本语言，它介于 Java 与 HTML 之间，以对象和事件驱动为基础。由于它的开发环境简单，不需要 Java 编译器就能直接在 Web 浏览器中运行，因此备受 Web 设计者们的青睐。

JavaScript 的历史

最初，Netscape 公司为了扩展浏览器的功能开发了一种名为 LiveScript 的脚本语言。1995 年 11 月末，Netscape 公司和 Sun 公司联合宣布把 LiveScript 改名为 JavaScript。

首先支持 JavaScript 标准的是 Netscape 公司制作的浏览器 Navigator 2.0 的 beta 3 版本。接着 Microsoft 公司的浏览器 Internet Explorer 3.0 也开始支持 JavaScript 了。此后，许多公司相继宣布承认 JavaScript 为 Internet 上的开放式脚本编写标准，并且把它添加到了自己的产品中。因此，目前流行的浏览器都支持 JavaScript。

JavaScript 的特性

JavaScript 是一种基于对象（Object）和事件驱动（Event Driven）的脚本语言，而且具有安全特性。将它与 HTML 语言和 Java 小程序组合在一起，可以把多个对象链接到一个页面中，从而实现与 Web 客户的交互。它是通过嵌入或调入到标准的 HTML 语言中运行的。它的出现弥补了 HTML 语言和 Java 的缺陷，是 Java 与 HTML 的好搭档。它具有以下几点特性。

脚本语言

JavaScript 是一种脚本语言，采用小程序段的方式来实现编程。与其他脚本语言一样，JavaScript 也是一种解释性执行的语言，它的开发过程简单易行。在句法构成上，JavaScript 的核心语言与 C、C++和 Java 相似，都具有诸如 if 语句、while 循环和&&运算符这样的结构成分。但是，JavaScript 与这些语言的相似之处也仅仅止于句法上的类同。它不像这些语言一样，需要预编

译再执行，而是在程序运行过程中被逐行解释的。与 HTML 标记结合在一起使用也大大的方便了用户的操作。

基于对象的语言

JavaScript 语言是基于对象的（Object-Based），而不是面向对象的（Object-Oriented）。之所以说它是一门基于对象的语言，主要是因为它没有提供诸如抽象、继承、重载等面向对象语言必备的特性，而是把其他语言所创建的复杂对象统一起来，形成了一个非常强大的对象系统。虽然 JavaScript 是一门基于对象的语言，但它还是具有面向对象语言的一些基本特性。它可以根据需要创建自己的对象，从而进一步扩大 JavaScript 的应用范围，增强 Web 文档的功能。

简单性

JavaScript 的简单性主要体现在它的语法结构是以 Java 的基本语句和控制流为基础的，所以熟悉 Java 的程序设计者，都不会对它的语法感到陌生。另外，它又不像 Java 那样是一种强类型的语言，也就是说它的变量不必具有一个明确的类型。

安全性

JavaScript 是一种具有安全性语言。它不允许客户访问本地的硬盘，也不允许把数据上载到服务器上，还不允许删除和修改网络文档，只允许用户在浏览器中浏览信息或进行动态的交互。这些措施有效地防止了数据的丢失。

交互性

JavaScript 的一个重要特性就是能够定义事件处理函数，即在特定的事件发生时要执行的任意代码段。这些事件通常都是用户触发的，例如把鼠标移动过一个超文本链接、在表单中输入了一个值或者点击了表单中的 Submit 按钮。具有图形界面（如 HTML 表单）的程序设计内在的要求一种事件驱动的模型，所以这种处理事件的能力是至关重要的。JavaScript 可以触发任意一种类型的动作来响应用户事件。

跨平台性

JavaScript 的解释执行只与浏览器本身有关，而与操作系统无关，所以只要你的计算机能够运行浏览器，而且该浏览器是 JavaScript 使能的，那么 JavaScript 脚本在其中就可以正确运行。

JavaScript 的语法

JavaScript 语言的句法结构与 Java 语言比较相似。由于 Java 语言来源于 C 语言，所以 JavaScript 的句法结构又与 C 语言比较相似。因此，如果熟悉 C、C++或 Java 这几种语言的程序设计者学习起 JavaScript 来会觉得游刃有余。不过，即使你从来没有接触过 C、C++或者 Java，学习起 JavaScript 来也不会感到很难，毕竟它比上述的程序设计语言要容易得多。

词法结构

JavaScript 是一种区分大小写的语言。这就是说，在输入语言的关键字、变量、函数名以及所有的标识符时，都必须采取一致的大小写形式。例如，关键字“for”就必须被输入为“for”，而不能输入为“For”或者“FOR”。同样的，“myname”、“Myname”、“MyName”和“MYNAME”是四个不同的变量名。

JavaScript 是忽略程序中记号之间的空格、制表符和换行符的，除非它们是字符串常量的一部分。所谓记号，就是一个关键字、变量名、数字、函数名等，显然你不会想在它们中间插入空格或换行符。如果你在一个记号中间插入了空格、制表符或换行符，那么就将它分成了两个记号，例如，myname 是一个记号，而 my name 则是两个独立的记号。

和 Java 一样，JavaScript 也支持 C++型的注释和 C 型的注释。JavaScript 会把处于“//”和一行结尾之间的任何文本都当作注释忽略掉。此外“/*”和“*/”之间的文本也将被当作注释。这些 C 型的注释可以跨越多行，但是其中不能有嵌套的注释。除了 C++型和 C 型的注释之外，浏览器端的 JavaScript 还能识别 HTML 注释的开始序列“<!--”。JavaScript 将把它看作单行注释，就像使用的是“//”注释一样。但是 JavaScript 不能识别 HTML 注释的结束序列“-->”。

在 JavaScript 中，标识符是用来命名变量和函数的。合法的标识符的命名规则和 Java 的相同。它的第一个字符必须是一个 ASCII 字母（小写或大写都可）、下划线（_）或者美元符号（$），接下来的字符可以是任意的字母、数字或下划线、美元符号。标识符不能和 JavaScript 中的关键字同名。

数据类型和特殊值

在 JavaScript 语言中，所有数据均具有某种数据类型。数据类型限定了数据的存储方式和可以对它们实施的操作。JavaScript 有三种基本数据类型——数字、字符串和布尔值。此外，它还支持两种复合数据类型——对象和数组，它们都是基本数据类型的集合。

数字

数字（number）是所有语言中最基本的数据类型。JavaScript 和其他程序设计语言（如 C 和 Java）的不同之处在于它并不区别整型数值和浮点型数值。在 JavaScript 中，所有的数值都是由浮点型表示的。它支持的数字形式有整数和浮点数。

整数是没有小数点的数，它可以表示成十进制、八进制和十六进制的形式。它们之间可以实现任意的转换。

八进制的数值是以数字 0 开头的，其后是一个数字序列，这个序列中的每个数字都在 0 和 7 之间（包括 0 和 7）。

十六进制的数值是以“0X”或者“0x”开头的，其后跟随的是十六进制的数字串。一个十六

进制的数字可能是 0 到 9 中的某个数字，也可能是 a（A）到 f（F）中的某个字母，它们是用来表示 0 到 15 之间（包括 0 和 15）的某个值的。例如，0xF 等于十进制中的 15，而 0x10 就等于十进制中的 16。

八进制数和十六进制数可以是负数，但不能是小数。一个以单个的“0”开头并包含小数点的数是十进制的浮点数。一个以“0x”或“00”开头并包含小数点的数其小数点右边的所有数都会被忽略。

浮点数是带小数点的数，它只能用十进制的形式表示。它有两种表示形式，分别为一般形式和指数形式（借助指数符号 e 或 E）：

- 一般形式：3.1415、4924.938
- 指数形式：3e10、4.319529E7

字符串

字符串是由字母、数字、标点符号等组成的序列，它是 JavaScript 用来表示文本的数据类型。程序中的字符串包含在单引号或双引号中。JavaScript 和 C、C++以及 Java 不同的是，它没有字符数据类型（如 char）。要表示单个的字符，必须使用长度为 1 的字符串。

在 JavaScript 的字符串中，反斜线符号具有特殊的用途。和 C 与 C++一样，在反斜线符号后加一个字符就可以表示在字符串中无法出现的字符。例如，\n 是一个转义序列，它表示的是一个换行符。表 1-1 列出了这些转义序列。

表1-1 JavaScript 的转义序列

序列	所代表的字符
\b	退格符
\f	换页符
\n	换行符
\r	回车符
\t	制表符
\’	撇号或单引号
\”	双引号
\\	反斜线符
\×××	一到三位八进制数说明的 Latin-1 编码标准中的字符。这个八进制数必须处于 0 到 377 之间，例如\374
\X××	两位十六进制数值说明的 Latin-1 编码标准中的字符。这个十六进制数必须处于 00 到 FF 之间，例如\x1A
\u××××	四位十六进制数说明的 Unicode 字符集中的字符。Navigator 4 不支持这种用法

布尔值

数字和字符串类型的值都是无穷多的，但是布尔数据类型则只有两个值，它们分别是“true”和“false”。一个布尔值代表一个“真值”，它说明了某个事物是真还是假。

布尔值通常用于 JavaScript 中的控制结构。例如，JavaScript 的 if/else 语句就是在布尔值为 true 时执行一个动作，而在布尔值为 false 时执行另一个动作。通常是将一个创建布尔值的比较与使用这个比较的语句结合在一起使用的。

对象

对象是数据段的集合。这些数据段通常是作为对象的属性来引用的。要引用一个对象的属性，采用的是在对象名后加点号和属性名的语法。例如，日期对象 date 有属性 year，要引用这个属性，使用的代码如下：

```
date.year; //date 是一个 Date 类型的对象，year 是该对象的一个属性
```

对象的属性在很多方面都与 JavaScript 的变量相似，它可以是任何类型的数据，包括数组、函数以及其他的对象。

如果一个函数值存储在某个对象的属性中，那么这个函数通常被称为方法，而属性名也就变成了方法名。要调用对象的一个方法，同样采用上述的语法将函数值从对象中提取出来，然后再使用调用函数的“()”语法即可。例如，日期对象 date 有方法 getYear()，要引用这个方法，使用的代码如下：

```
date.getYear(); //date 是一个 Date 类型的对象，getYear()是它的一个方法
```

数组

数组和对象一样是数据的集合。它们的不同之处在于对象中的数据都是由名字来标识的，而数组的数据则是由下标来标识的。在 JavaScript 中，要访问数组中的某个数据，可以使用数组名加下标的语法，其中下标是用方括号括起来的。例如，如果一个数组名为 a，i 是一个非负整数，那么 a[i]就是一个数组元素。因为数组的下标值是从 0 开始的，所以 a[2]引用的是数组 a 的第三个元素。数组的元素是任意类型的。你可以使用[]运算符来访问数组的元素，在方括号左边是数组名，方括号之中是任意一个具有非负整数值的表达式。

在像 C 和 Java 这样的语言中，数组是具有固定的元素个数的，你必须在创建这个数组的时候就指定它的元素数。而在 JavaScript 则不是这样，它的数组可以具有任意个数的元素，你可以在任何时刻改变这个元素数。要给一个数组添加新的元素，只需要给它赋一个值即可。

特殊值

JavaScript 的关键字 null 是一个特殊的值，它表示的是“无值”。null 常常被看作是特殊数据类型 Null 的惟一合法值。有时它则被看作对象类型的一个特殊值，即代表“无对象”。无论在哪种情况下，null 都是个惟一值，有别于所有其他的值。如果一个变量的值为 null，那么它就不是有效的对象、数组、数字、字符串或布尔值。

还有一种特殊的 JavaScript 值是 undefined。在你使用了一个并不存在的变量时，或者使用了已经声明但还没有赋值的变量时，又或者使用了一个并不存在的对象属性时，得到的就是这个值。和 null 不同，未定义的值并没有关键字 undefined。

变量

所谓变量，就是内存中用标识符命名的存储单元，用于存放可由应用程序修改的数据。命名变量的标识符称为该变量的变量名。JavaScript 和 Java 与 C 这样的语言之间存在一个重要的差别，那就是 JavaScript 是无类型的。也就是说，JavaScript 的变量可以存放任何数据类型的值，而 Java 和 C 的变量都只能存放指定类型的数据。有了这种特性，JavaScript 就能在必要的时候迅速地把一种类型的数据转换成另一种类型的。

变量的定义

定义变量的方式有两种。一种是通过直接赋值来定义变量，此时 JavaScript 会自动地用那个变量名为你创建一个全局变量的。另一种方法是使用关键字 var 来定义变量。如果你想在函数内部创建一个局部变量，那么最好使用 var 语句来定义变量。否则你使用的可能是一个已经存在了的全局变量。

- 直接赋值：x=16
- 关键字赋值：var x=7

使用 var 语句既可以一次定义多个变量，也可以重复定义一个变量，而且在定义变量的同时，它还能够初始化所定义的变量的值。由 var 语句定义的变量是永久性的，也就是说，使用 delete 运算符删除这种变量会引发错误的。

变量的作用域

一个变量的作用域是程序中定义这个变量的区域。全局变量的作用域是全局性的，即在 JavaScript 代码中，它处处都有定义。而在函数之内声明的变量是局部变量，它们只在函数体内部有定义，其作用域是局部性的。函数的参数也是局部变量，它们只在函数体内部有定义。

在函数体内部，局部变量的优先级比同名的全局变量高。如果你定义的局部变量或函数的参数与某个全局变量同名，那么就会把这个全局变量隐藏起来了。

变量与对象的属性

你一定已经注意到了，变量和对象的属性之间有许多相似之处。事实上，变量也是一个对象的属性，这个对象就称为全局对象。当 JavaScript 的解释程序开始运行时，它首先要做的事情之一就是在执行任何 JavaScript 代码之前，创建一个全局对象。你在定义一个 JavaScript 的全局变量时，实际上所做的是在定义全局对象的一个属性。在浏览器端的 JavaScript 代码中，代表浏览器窗口的 Window 对象就是作为全局对象的。

如果全局变量是全局对象的属性，那么局部变量又是什么呢？它们同样是一个对象的属性，这个对象被称为调用对象（call object）。虽然调用对象的生命周期比全局对象的短，但是它们的用途是相同的。在执行一个函数时，函数的参数和局部变量是作为调用对象的属性的。用一个完全独立的对象来存储局部变量可以防止局部变量覆盖同名的全局变量。

表达式和运算符

表达式是 JavaScript 的一个“短语”，JavaScript 的解释程序对可以计算它，从而生成一个值。最简单的表达式就是常量或变量名。常量表达式的值就是这个常量本身。变量表达式的值则是该变量所引用的值。此外，还可以通过合并简单的表达式来创建较为复杂的表达式。

连接表达式的是运算符。由此可见，表达式需要相应的运算符来实现，有什么样的表达式就需要什么样的运算符。我们可以把表达式按运算数的类型划分为 5 种，同时也就有了 5 种相关的运算符。JavaScript 的表达式如下：

- 算术表达式：通过算术运算符完成数的运算
- 赋值表达式：通过赋值运算符完成变量的赋值
- 字符串表达式：通过字符串运算符完成字符串的运算
- 逻辑表达式：通过逻辑运算符完成布尔值的运算
- 条件表达式：通过条件运算符完成运算

算术运算符及表达式

JavaScript 语言中的算术运算符包括一元运算符和二元运算符：

- 一元运算符：-、++、--
- 二元运算符：+、-、*、/、%

以上一元运算符都是左结合的。“-”的功能是对变量取负。“++”、“--”是对变量进行加一或减一运算，即 i ++或++ i 等价于 i = i + 1，i --或-- i 等价于 i = i - 1。二元运算符都是右结合的，“+”是实现加法运算的，“-”是实现减法运算的，“*”是实现乘法运算的，“/”是实现除法运算的，它返回的结果是运算后得到的商值，“%”也是实现除法运算的，不过它返回的是运算后得到的余数。

i++与++i 的区别在于 i++是先使用 i 值，然后再进行加法运算，而++i 是先执行加法运算，然后再进行其他运算。下面举例说明：

```
int n=6;
int m=(n++)*3;
//执行的结果是 n=7,m=18;
int n=6;
int m=(++n)*3;
//执行的结果是 n=7,m=21;
//i--与--i 的区别同理。
```

赋值运算符及表达式

用赋值运算符连接起来的表达式叫做赋值表达式。所使用的赋值运算符如表 1-2 所示。

表1-2 赋值运算符

运算符	所表示的含义
=	右运算数直接赋值给左运算数
+=	右运算数加左运算数后，和赋值给左运算数
-=	右运算数减左运算数后，差赋值给左运算数
*=	右运算数乘左运算数后，积赋值给左运算数
/=	右运算数除左运算数后，商赋值给左运算数
%=	右运算数除左运算数后，余数赋值给左运算数

赋值表达式的常用格式如下所示：

```
variable = expression
```

其中 variable 表示的是一个变量，expression 表示的是一个表达式。

字符串运算符及表达式

JavaScript 语言中的字符串运算符主要包括连接运算符和比较运算符：

- 连接运算符：+
- 比较运算符：<、>、==、<=、>=、!=

连接运算符用于连接两个字符串，其基本格式为：

```
返回值=字符串 1+字符串 2
```

例如，“girl”+“friend”的返回值为“girlfriend”。如果变量 x 表示字符串“boy”，x+=“friend”的返回的就是 x=boyfriend。

比较运算符是比较两个字符串的相应 ASCII 码大小的。它从第一个字符开始依次比较相应字符的 ASCII 码值，遇到不同的值时，就停止比较，返回比较结果。只有两个字符串中的字符都相同时，两个字符串才相等。如果字符串的长度不相同，字符较少的字符串就小于字符较多的字符串。

逻辑运算符及表达式

JavaScript 语言中的逻辑运算符如下：

- 逻辑运算符：！（一元逻辑非）、&&（二元逻辑与）、‖（二元逻辑或）

运算符！是一个一元运算符，被放在一个运算数之前。它的目的是对运算数的布尔值取反。

当运算符&&的两个运算数都是布尔值时，它对这两个运算数执行布尔 AND 操作，即当且仅