第 1 章

JavaScript 基础

本章技能目标

- 掌握 JavaScript 的基本语法
- 掌握选择结构之 if 语句的用法

本章简介

通过对网页制作基础课程的学习，我们对网站的制作已经有了比较深刻的理解，知道如何制作一个网页、如何根据需求搭建一个网站，但是如何让网页更加绚丽多彩、如何增加用户的良好体验，这些还是我们前端开发人员努力的方向。本章开始我们进入网页特效方面的学习。

本章主要介绍为什么要学习 JavaScript，JavaScript 的基本语法结构，如何在网页中应用 JavaScript 等，有了这些基本技能，后续我们就可以进入真正的特效制作了。

1 JavaScript 概述

1.1 JavaScript 概念

为什么学习 JavaScript？主要基于以下两点原因。

1. 客户端表单验证，减轻服务器压力

网站中常见会员注册页面，我们填写注册信息时，如果某项信息格式输入错误（例如：密码长度位数不够等），表单页面将及时给出错误提示。这些错误在没有提交到服务器前，由客户端提前进行验证，称为客户端表单验证。这样，用户得到了即时的交互（反馈填写情况），同时也减轻了网站服务器端的压力，这是 JavaScript 最常用的场合。

2. 制作页面动态特效

在 JavaScript 中，可以编写响应鼠标单击等事件的代码，创建动态页面特效，从而高效地控制页面的内容。例如，表单的验证效果（如图 1.1 所示），或者网页轮播效果特效（如图 1.2 所示）等，它们可以在有限的页面空间里展现更多的内容，从而增加客户端的体验，进而使我们的网站更加有动感、有魅力，吸引更多的浏览者。

图 1.1 表单验证

图 1.2 轮播效果

这里要说明一点，虽然 JavaScript 可以实现许多动态效果，但要实现一个特效可能需要十几行，甚至几十行，而使用 jQuery（JavaScript 程序库）可能只需要几行就可以实现同样的效果，关于 jQuery 方面的技术，会在后面章节讲解，而 JavaScript 是学习 jQuery 的基础，打好基础至关重要。

1.2 JavaScript 的应用

那么到底什么是 JavaScript 呢？JavaScript 是一种描述性语言，也是一种基于对象（Object）和事件驱动（Event Driven）的、并具有安全性能的脚本语言。它与 HTML 超文本标记语言一起，在一个 Web 页面中链接多个对象，与 Web 客户实现交互。无论在客户端还是在服务器端，

JavaScript 应用程序都要下载到浏览器的客户端执行，从而减轻了服务器端的负担。总结其特点如下：

- JavaScript 主要用来向 HTML 页面中添加交互行为。
- JavaScript 是一种脚本语言，语法和 Java 类似。
- JavaScript 一般用来编写客户端脚本。
- JavaScript 是一种解释性语言，边执行边解释。

1.2.1 JavaScript 的组成

一个完整的 JavaScript 是由以下三个不同的部分组成的，如图 1.3 所示。

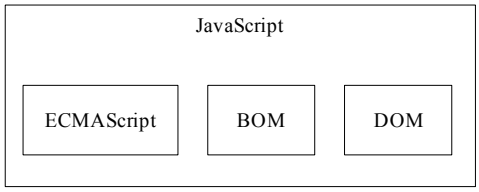

图 1.3 JavaScript 的组成

1. ECMAScript 标准

ECMAScript 是一种开放的、国际上广为接受的、标准的脚本语言规范。它不与任何具体的浏览器绑定。ECMAScript 标准主要描述了以下内容。

- 语法。
- 变量和数据类型。
- 运算符。
- 逻辑控制语句。
- 关键字、保留字。
- 对象。

ECMAScript 是一个描述，规定了脚本语言的所有属性、方法和对象的标准，因此在使用 Web 客户端脚本语言编码时一定要遵循 ECMAScript 标准。

2. 浏览器对象模型（BOM）

BOM 是 Browser Object Model（浏览器对象模型）的简称，提供了独立于内容与浏览器窗口进行交互的对象，使用浏览器对象模型可以实现与 HTML 的交互。

3. 文档对象模型（DOM）

DOM 是 Document Object Model（文档对象模型）的简称，是 HTML 文档对象模型（HTML DOM）定义的一套标准方法，用来访问和操纵 HTML 文档。

关于 BOM 和 DOM 的内容将在后面章节中讲解，本章着重讲解 ECMAScript 标准。

1.2.2 JavaScript 的执行原理

了解了 JavaScript 的组成，下面再来深入了解 JavaScript 脚本语言的执行原理。

在脚本的执行过程中，浏览器客户端与应用服务器端采用请求/响应模式进行交互，如图 1.4 所示。

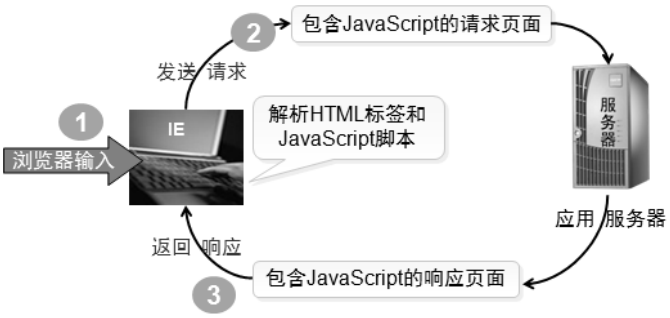

图 1.4　脚本执行原理

下面我们逐步分解一下这个过程：

（1）浏览器向服务器端发送请求：一个用户在浏览器的地址栏中输入要访问的页面（页面中包含 JavaScript 脚本程序）。

（2）数据处理：服务器端将某个包含 JavaScript 脚本的页面进行处理。

（3）发送响应：服务器端将含有 JavaScript 脚本的 HTML 文件处理页面发送到浏览器客户端，然后由浏览器从上至下逐条解析 HTML 标签和 JavaScript 脚本，并将页面效果呈现给用户。

使用客户端脚本的好处有以下两点：

- 含脚本的页面只要下载一次即可，这样能减少不必要的网络通信。
- 脚本程序是由浏览器客户端执行，而不是由服务器端执行，因此能减轻服务器端的压力。

1.2.3　JavaScript 的基本结构

通常，JavaScript 代码是用<script>标签嵌入 HTML 文档中的。可以将多个脚本嵌入到一个文档中，只需将每个脚本都封装在<script>标签中即可。浏览器在遇到 <script> 标签时，将逐行读取内容，直到遇到</script> 结束标签为止。然后，浏览器将检查 JavaScript 语句的语法，如果有任何错误，就会在警告框中显示；如果没有错误，浏览器将编译并执行语句。

脚本的基本结构如下：

```
<script type="text/javascript">
    <!--
        JavaScript 语句;
    -->
</script >
```

type 是<script>标签的属性，用于指定文本使用的语言类别为 JavaScript。

<!--语句-->是注释标签。这些标签用于告知不支持 JavaScript 的浏览器忽略标签中包含的语句。<!--表示开始注释标签，-->则表示结束注释标签。这些标签是可选的，但最好在脚本中使用这些标签。目前大多数浏览器支持 JavaScript，但使用注释标签可以确保不支持 JavaScript 的浏览器会忽略嵌入到 HTML 文档中的 JavaScript 语句。

注意：有的网页中用 language = "javascript" 来表示使用的语言是 JavaScript，因为 XHTML1.0 已明确表示不支持这种写法，所以这种写法不推荐。

下面通过一个示例来深入学习脚本的基本结构，代码如示例 1 所示。

示例 1

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>输出 Hello World</title>
<script type="text/javascript">
<!--
      document.write("使用 JavaScript 脚本输出 Hello World");
      document.write("<h1>Hello World</h1>");
-->
</script>
</head>
<body>
</body>
</html>
```

示例 1 在浏览器中的运行效果如图 1.5 所示。

图 1.5　使用 JavaScript 脚本输出 Hello World

代码中，document.write()是用来向页面输出可以包含 HTML 标签的内容。把 document.write()语句包含在<script>与</script>之间，浏览器就会把它当作一条 JavaScript 命令来执行，这样浏览器就会向页面输出内容。

经验：

如果不使用<script>标签，浏览器就会将 document.write("<h1>Hello World</h1>")当作是纯文本来处理，也就是说会把这条命令本身写到页面上。

<script>…</script>的位置并不是固定的，可以包含在文档中的任何地方，只要保证这些代码在被使用前已读取并加载到内存即可。

1.3　在网页中引用 JavaScript

学习了脚本的基本结构和脚本的执行原理，如何在网页中引用 JavaScript 呢？JavaScript 作为客户端程序，嵌入网页有以下三种方式：

- 使用<script>标签。
- 使用外部 JavaScript 文件。

- 直接嵌入在 HTML 标签中。

1. 使用<script>标签

示例 1 就是直接使用<script>标签将 JavaScript 代码加入到 HTML 文档中。这是最常用的方法，但这种方式通常适用于 JavaScript 代码较少，并且网站中的每个页面使用的 JavaScript 代码均不相同的情况。

2. 使用外部 JavaScript 文件

在实际工作中，有时会希望在若干个页面中运行 JavaScript 实现相同的页面效果，针对这种情况，通常使用外部 JavaScript 文件。外部 JavaScript 文件是将 JavaScript 写入一个外部文件中，以*.js 为后缀保存，然后该文件指定给<script>标签中的“src”属性，这样就可以使用这个外部文件了。这种方式与在网页中引用外部样式类似。示例 1 中实现的页面效果若使用外部 JavaScript 文件实现如示例 2 所示。

示例 2

hello.js 文件代码：

```
document.write("使用 JavaScript 脚本输出 Hello World");
document.write("<h1>Hello World</h1>");
```

export.html 页面代码：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>输出 Hello World</title>
<script src="hello.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

hello.js 就是外部 JavaScript 文件，src 属性表示指定外部 JavaScript 文件的路径，在浏览器中运行示例 2，运行的结果与示例 1 的运行结果一样。

注意：外部文件不能包含<script>标签，通常将.js 文件放到网站目录中单独存放脚本的子目录中（一般为 js），这样容易管理和维护。

3. 直接嵌入在 HTML 标签中

有时需要在页面中加入简短的 JavaScript 代码实现一个简单的页面效果，例如单击按钮时弹出一个对话框等，这样通常会在按钮事件中加入 JavaScript 处理程序。下面的例子就是单击按钮弹出消息框。

关键代码如下所示：

```
<input name="btn" type="button" value="弹出消息框" onclick="javascript:alert('欢迎你');"/>
```

当单击“弹出消息框”按钮时，弹出提示对话框，如图 1.6 所示。

代码中，onclick 是单击的事件处理程序，当用户单击按钮时，就会执行“javascript:”后面的 JavaScript 命令，alert()是一个功能函数，作用是向页面弹出一个对话框。

图 1.6　提示对话框

操作案例 1：网页中引用 JavaScript 代码

需求描述

利用在页面中直接添加 JavaScript 代码的方式，完成图 1.6 提示对话框的显示。

完成效果

打开网页，单击按钮，弹出提示框，见图 1.6。

技能要点

在网页中引用 JavaScript。

实现思路

网页中添加<input>标签，利用 onclick 属性指定弹出提示内容。

2　JavaScript 基础语法

JavaScript 像 Java、C#一样，也是一门编程语言，也包含变量的声明、赋值、运算符号、逻辑控制语句等基本语法，下面我们就来学习 JavaScript 的基本语法。

2.1　变量

JavaScript 是一种弱类型语言，没有明确的数据类型，也就是说，在声明变量时，不需要指定变量的类型，变量的类型由赋给变量的值决定。

在 JavaScript 中，变量是使用关键字 var 声明的。下面是 JavaScript 声明变量的语法格式。

```
var  合法的变量名;
```

其中，var 是声明变量所使用的关键字；“合法的变量名”是遵循 JavaScript 中变量命名规则的变量名。JavaScript 中的变量命名可以由数字、字母、下划线和“$”符号组成，但首字母不能是数字，并且不能使用关键字命名。为变量赋值有三种方法：

- 先声明变量再赋值。
- 同时声明和赋值变量。
- 不声明直接赋值。

例如声明变量的同时为变量赋值：

```
var width = 20;          //在声明变量 width 的同时，将数值 20 赋给了变量 width
var x, y, z = 10;        //在一行代码中声明多个变量时，各变量之间用逗号分隔
```

不声明变量而直接使用：

```
x=88;  //没有声明变量 x，直接使用
```

注意：

JavaScript 区分大小写，特别是变量的命名、语句关键字等，这种错误有时很难查找。

变量可以不经声明而直接使用，但这种方法很容易出错，也很难查找排错，不推荐使用。在使用变量之前，应先声明后使用，养成良好的编程习惯。

2.2 数据类型

尽管在声明变量时不需要声明变量的数据类型，而是由赋给变量的值决定。但在 JavaScript 中，提供了常用的基本数据类型，这些数据类型如下所示：

- undefined（未定义类型）。
- null（空类型）。
- number（数值类型）。
- string（字符串类型）。
- boolean（布尔类型）。

1. undefined 类型

如前面的示例显示的一样，undefined 类型只有一个值，即 undefined。当声明的变量未初始化时，该变量的默认值是 undefined。

```
var width;
```

这行代码声明了变量 width，且此变量没有初始值，将被赋予值 undefined。

2. null 类型

只有一个值的类型是 null，是一个表示“什么都没有”的占位符，可以用来检测某个变量是否被赋值。值 undefined 实际上是值 null 派生来的，因此 JavaScript 把它们定义为相等的。

```
alert(null==undefined);    //返回值为 true
```

尽管这两个值相等，但它们的含义不同，undefined 是声明了变量但未对该量赋值，null 则表示对该变量赋予了一个空值。

3. number 类型

JavaScript 中定义的最特殊的类型是 number 类型，这种类型既可以表示 32 位的整数，还可以表示 64 位的浮点数。下面的代码声明了存放整数值和浮点数值的变量。

```
var iNum=23;
var iNum=23.0;
```

整数也可以表示为八进制或十六进制，八进制首数字必须是 0，其后的数字可以是任何八进制数字（0～7），十六进制首数字也必须是 0，后面是任意的十六进制数字和字母（0～9 和 A～F），例如下面的代码：

```
var iNum=070;          //070 等于十进制的 56
var iNum=0x1f;         //0x1f 等于十进制的 31
```

对于非常大或非常小的数，可以用科学计数法表示浮点数，也是 number 类型。另外一个特殊值 NaN（Not a Number）表示非数，它是 number 类型，例如：

```
typeof(NaN);           //返回值为 number
```

4. string 类型

（1）字符串定义。

在 JavaScript 中，字符串是一组被引号（单引号或双引号）括起来的文本，例如：

```
var string1="This is a string";      //定义了一个字符串 string1
```

（2）字符的属性与方法。

JavaScript 语言中的 String 也是一种对象，它有一个 length 属性，表示字符串的长度（包括空格等），调用 length 的语法如下：

```
字符串对象.length;
```

在 JavaScript 中，字符串对象的使用语法如下：

```
字符串对象.方法名();
```

JavaScript 语言中的 String 对象也有许多方法用来处理和操作字符串，常用的方法如表 1-1 所示。

表 1-1　String 对象常用方法

方法	描述
toString()	返回字符串
toLowerCase()	把字符串转化为小写
toUpperCase()	把字符串转化为大写
charAt(index)	返回在指定位置的字符
indexOf(str,index)	查找某个指定的字符串在字符串中首次出现的位置
substring(index1,index2)	返回位于指定索引 index1 和 index2 之间的字符串，并且包括索引 index1 对应的字符，不包括索引 index2 对应的字符
split(str)	将字符串分割为字符串数组

5. boolean 类型

boolean 型数据被称为布尔型数据或逻辑型数据，boolean 类型是 JavaScript 中最常用的类型之一，它只有两个值 true 和 false。

有时候需要检测变量的具体数据类型，JavaScript 提供了 typeof 运算符来判断一个值或变量究竟属于哪种数据类型。语法为：

```
typeof(变量或值)
```

其返回结果有以下几种：

- undefined：如果变量是 undefined 型的。
- number：如果变量是 number 型的。
- string：如果变量是 string 型的。
- boolean：如果变量是 boolean 型的。
- object：如果变量是 null 型，或者变量是一种引用类型，例如对象、函数、数组。

例如，如下示例将在页面输出“name:string”：

```
var name="rose";
document.write("name: "+typeof(name));
```

2.3 运算符

在 JavaScript 中常用的运算符可分为算术运算符、赋值运算符、比较运算符和逻辑运算符，如表 1-2 所示。

表 1-2 常用的运算符

类别	运算符号
算术运算符	+、-、*、/、%、++、--
赋值运算符	=
比较运算符	>、<、>=、<=、==、!=
逻辑运算符	&&、||、！

1. 算术运算符

算术运算符用于执行变量与/或值之间的算术运算，如加（+）、减（-）、取余（%）等。例如：

```
var x=5;
var y=x%2;      //y 的值为 1
```

2. 赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

3. 比较运算符

比较运算符在逻辑语句中使用，以测定变量或值之间的关系，如大于（>）、小于等于（<=）、等于（==）、不等于（!=）。

4. 逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑关系。

2.4 注释

注释是描述部分程序功能或整个程序功能的一段说明性文字，注释不会被解释器执行，而是直接跳过。注释的功能是帮助开发人员阅读、理解、维护和调试程序。JavaScript 语言的注释与 Java 语言的注释一样，分为单行注释和多行注释两种。

- 单行注释以 // 开始，以行末结束，例如：

```
alert("恭喜你!注册会员成功");    //在页面上弹出注册会员成功的提示框
```

- 多行注释以 /* 开始，以 */ 结束，例如：

```
/*
在页面上输出 5 次"Hello World"
*/
for(var i=0;i<5;i++){
    document.write("<h3>Hello World</h3>");
}
```

2.5 选择结构

程序简单讲就是一系列有序指令的集合，使用逻辑控制语句控制程序的执行顺序。程序结构主要分为三大类：顺序结构、选择结构和循环结构。本章中，我们首先简单了解一下选择结构。

选择结构（有时也称为条件结构），就是基于不同的条件来执行不同的动作，实现不同的结果。选择结构分为 if 结构和 switch 结构这两种，下面详细介绍选择结构。

1. 选择结构之基本 if 结构

基本语法如下：

```
if(表达式){
        //JavaScript 语句 1;
 }
```

其中，当表达式的值为 true 时，才执行 JavaScript 语句 1。

2. 选择结构之 if…else 结构

基本语法如下：

```
if(表达式){
        //JavaScript 语句 1;
 }else{
        //JavaScript 语句 2;
 }
```

其中，当表达式的值为 true 时，执行 JavaScript 语句 1，否则执行后面的语句 2。

3. 选择结构之多重 if 结构

基本语法如下：

```
if(表达式 1) {
        //JavaScript 语句 1
}
else if(表达式 2) {
        //JavaScript 语句 2
}
else {
        //JavaScript 语句 3
}
```

其中，当表达式 1 的值为 true 时，执行 JavaScript 语句 1，否则进行再判断，判断表达式 2 如果为 true，执行语句 2，否则执行语句 3。

4. 选择结构之 switch 结构

基本语法如下：

```
switch(表达式){
    case 值 1:
      //JavaScript 语句 1;
      break;
    case 值 2:
```

```
        //JavaScript 语句 2;
          break;
        …
        default:
          //JavaScript 语句 n;
          break;
    }
```

其中，case 表示条件判断，关键字 break 会使代码跳出 switch 语句，如果没有关键字 break，代码就会继续执行，进入下一个 case。关键字 default 说明表达式的结果不等于任何一种情况。

在 JavaScript 中，switch 语句可以用于数值和字符串，例如：

```
var weekday="星期一";
switch(weekday){
      case "星期一":
        alert("新的一周开始了");
        break;
      case "星期五":
        alert("明天就可以休息了");
        break;
      default:
        alert("努力工作");
        break;
      }
```

2.6 信息提示的使用

在网上冲浪时，页面上经常会弹出一些信息提示框，例如注册时弹出提示输入信息的提示框，或者弹出一个等待用户输入数据的对话框等，这样的输入或输出在 JavaScript 中称为警告对话框（alert）和提示对话框（prompt）。

1. 警告（alert）

alert()方法前面已经用过，此方法会创建一个特殊的小窗口，该窗口带有一个字符串和一个“确定”按钮，如图 1.7 所示。

图 1.7 警告对话框

alert()方法的基本语法格式为：

```
alert("提示信息");
```

该方法将弹出一个警告对话框，其内容可以是一个变量的值，也可以是一个表达式的值。如果要显示其他类型的值，需要将其强制转换为字符串型。以下代码都是合法的：

```
var userName="rose";
var string1="我的名字叫 rose";
alert("Hello World");
alert("我的名字叫"+userName);
alert(string1);
```

2. 提示（prompt）

prompt()方法会弹出一个提示对话框，等待用户输入一行数据。

prompt()方法的基本语法格式为：

```
prompt("提示信息","输入框的默认信息");
```

注意：程序调试是 JavaScript 中的一个重要环节，在 JavaScript 中 alert()方法经常被用来进行调试程序。通过 alert()方法将不确定的数据以信息框的方式展示，以此来判断出现错误的位置。

操作案例 2：模拟计算器

需求描述

实现一个简易版计算器功能，要求如下：

- 用户分别输入两个数字以及运算符。
- 给出运算结果。

完成效果

用户输入效果如图 1.8 所示，给出运算结果界面如图 1.9 所示。

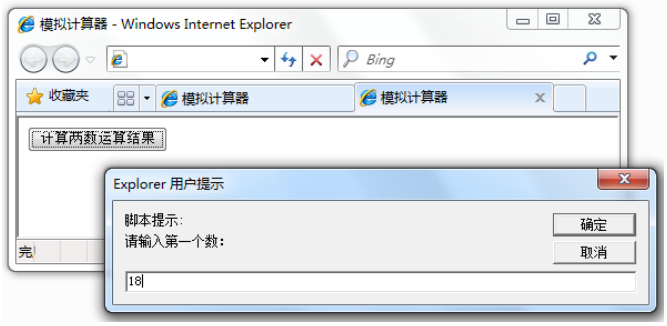

图 1.8　输入数字及运算符

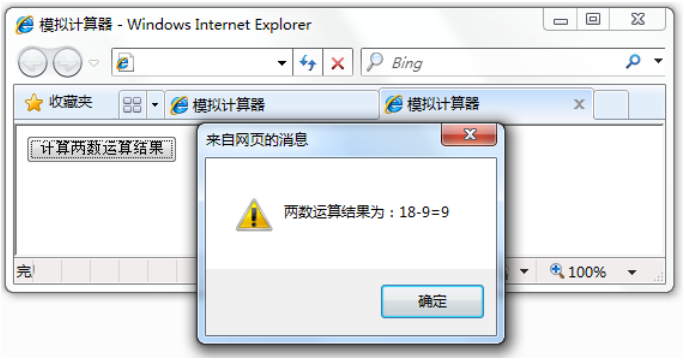

图 1.9　运算结果

技能要点

- 多重 if 结构。
- JavaScript 中称为警告对话框（alert）和提示对话框（prompt）。

关键代码

- 使用 prompt()方法获取两个变量的值和一个运算符。代码如下：

```
var op1=prompt("请输入第一个数：","");
var op2=prompt("请输入第二个数：","");
var sign=prompt("请输入运算符号","")
var result;
opp1=parseFloat(op1);
opp2=parseFloat(op2);
```

- 运算结果使用 alert()方法显示出来。代码如下：

```
alert("两数运算结果为："+op1+sign+op2+"="+result);
```

3 函数

在 JavaScript 中，函数是程序的基本单元，是完成特定任务的代码语句块，执行特定的功能。

JavaScript 中的函数有两种：一种是 JavaScript 自带的系统函数，另一种是用户自行创建的自定义函数，下面分别来学习这两种函数。

3.1 系统函数

JavaScript 提供了许多系统函数供开发人员使用，这些系统函数已经实现了某些功能，开发人员直接调用就可以了。后续的学习中，我们会接触到很多系统函数，后续章节会进行详细讲解，下面举例介绍几个比较常用的系统函数。

1. parseInt()与 parseFloat()

parseInt()函数可解析一个字符串，并返回一个整数，语法格式为：

```
parseInt("字数串")
```

例如：

```
var num1=parseInt("78.89")          //返回值为 78
var num2=parseInt("4567color")      //返回值为 4567
var num3=parseInt("this36")         //返回值为 NaN
```

parseFloat()函数可解析一个字符串，并返回一个浮点数，语法格式为：

```
parseFloat("字数串")
```

parseFloat()函数与 parseInt()函数的处理方式相似，从位置 0 开始查看每个字符，直到找到第一个非有效的字符为止，然后把该字符之前的字符串转换为浮点数。

对于这个函数来说，第一个出现的小数和点是有效字符，如果有两个小数点，那么第二个小数点被看作是无效的。例如：

```
var num1=parseFloat("4567color");      //返回值为 4567
var num1=parseFloat("45.58");          //返回值为 45.58
var num1=parseFloat("45.58.25");       //返回值为 45.58
```

```
var num1=parseFloat("color4567");      //返回值为 NaN
```

2. isNaN()

isNaN()函数用于检查其参数是否是非数字，语法格式为：

```
isNaN(x)
```

如果 x 是特殊的非数字值，返回值就是 true，否则返回 false。例如：

```
var flag1=isNaN("12.5");                    //返回值为 false
var flag2=isNaN("12.5s");                   //返回值为 true
var flag3=isNaN(45.8);                      //返回值为 false
```

注意：isNaN()函数通常用于检测 parseFloat()和 parseInt()的结果，以判断它们表示的是否为合法的数字。也可以用 isNaN()函数来检测算数的错误，例如用 0 作除数的情况。

3.2 自定义函数

自定义函数指的是开发人员根据业务需求，自行开发的功能代码块。

同任何一种编程语言一样，JavaScript 中自定义函数也需要先定义函数，然后才能调用函数。下面学习如何定义函数及调用函数。

1. 定义函数

在 JavaScript 中，自定义函数由关键字 function、函数名、一组参数以及置于括号中的待执行的 JavaScript 语句组成，语法格式为：

```
function  函数名(参数 1,参数 2,参数 3,…){
//JavaScript 语句;
[return  返回值]
}
```

function 是定义函数的关键字，必须得有。

参数 1、参数 2 等是函数的参数。因为 JavaScript 本身是弱类型，所以它的参数也没有类型检查和类型限定。函数中的参数是可选的，根据函数是否带参数，可分为不带参数的无参函数和有参函数。例如：

```
function  函数名(){
   //JavaScript 语句;
}
```

“{”和“}”定义了函数的开始和结束。

return 语句用来规定函数返回的值。

2. 调用函数

要执行一个函数，必须先调用这个函数，当调用函数时，必须指定函数名及其后面的参数（如果有参数）。函数的调用一般和元素的事件结合使用，调用格式如下：

```
事件名="函数名()";
```

下面通过示例 3 和示例 4 来学习如何定义函数和调用函数。

示例 3

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

```
function showHello(){
        for(var i=0;i<5;i++){
                document.write("<h2>Hello World</h2>");
            }
 }
-->
</script>
…
<input name="btn" type="button" value="显示 5 次 HelloWorld" onclick= "showHello()" />
```

showHello()是创建的无参函数。

onclick 表示按钮的单击事件，当单击按钮时调用函数 showHello()。

在浏览器中运行示例 7，如图 1.10 所示，单击“显示 5 次 HelloWorld”按钮，调用无参函数 showHello()，在页面中循环输出 5 行“Hello World”。

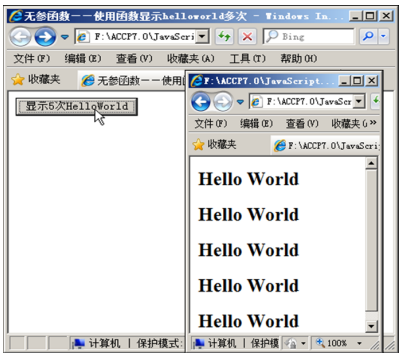

图 1.10　调用无参函数

在示例 3 中使用的是无参函数，运行一次页面只能输出 5 行“Hello World”，如果需要根据用户的要求每次输出不同行数，该怎么办呢？有参函数可以实现这样的功能。

下面修改示例 3，把函数 showHello()修改成一个有参函数，使用 prompt()提示用户每次输出“Hello World”的行数，然后将 prompt()方法返回的值作为参数传递给函数 showHello()。

示例 4

```
<script type="text/javascript">
<!--
function showHello(count){
      for(var i=0;i<count;i++){
              document.write("<h2>Hello World</h2>");
            }
}
-->
</script>
…
<input name="btn" type="button" value="请输入显示 HelloWorld 的次数" onclick="showHello(prompt('请输入显示 Hello World 的次数:',''))"/>
…
```

count 表示传递的参数，不需要定义数据类型。

将 prompt()方法返回的值作为参数传递给函数 showHello(count)。

在浏览器中运行示例 4，单击页面上的按钮，弹出提示用户输入显示 Hello World 次数的窗口，用户输入值后，根据用户输入的值在页面上输出 Hello World，如图 1.11 所示。

图 1.11　动态显示 Hello World

3.3　变量的作用域

在 JavaScript 中，根据变量作用范围不同，可分为全局变量和局部变量。

JavaScript 中的全局变量，是在所有函数之外的脚本中声明的变量，作用范围是该变量定义后的所有语句，包括其后定义的函数中的代码，以及其后的<script>与</script>标签中的代码，例如下面的示例代码中声明的变量 i。

JavaScript 中的局部变量，是在函数内声明的变量，只有在该函数中且位于该变量之后的代码可以使用这个变量，如果在之后的其他函数中声明了与这个局部变量同名的变量，则后声明的变量与这个局部变量毫无关系。

请使用断点调试的方式运行下面的示例 5，分析全局变量和局部变量的作用。

示例 5

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>变量的作用范围</title>
<script type="text/javascript">
<!--
var i=20;
function first(){
    var i=5;
    for(var j=0;j<i;j++){
        document.write("    "+j);
    }
}
```

```
function second(){
    var t=prompt("输入一个数","")
    if(t>i){
        document.write(t);}
    else{
       document.write(i);}
    first();
}
-->
</script>
</head>
<body onload="second()">
</body>
</html>
```

运行上面的例子，在 prompt()弹出的输入框中输入 12，单击“确定”按钮，运行效果如图 1.12 所示。

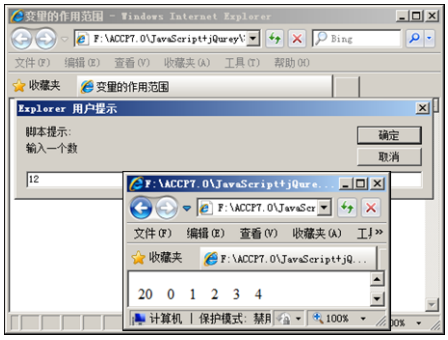

图 1.12　变量的作用范围

这里使用了 onload 事件，onload 事件会在页面加载完成后立即发生。将断点设置在 var i=20;这一行，按单步运行。我们会发现，先执行 var i=20，设置 i 为全局变量，接着运行 onload 事件调用 second()函数，在函数中，因为输入的值 12 小于 20，因此执行 else 语句，即在页面中输出了 20。然后执行函数 first()，在函数 first()中，声明的 i 为局部变量，它只作用于函数 first()中，因此 for 循环输出了 0、1、2、3、4。

操作案例 3：模拟 QQ 登录验证

需求描述

QQ 登录过程中，验证用户输入的用户名和密码，要求如下：

- 与事先设定好的用户名（例如 hello）、密码（例如 123456）进行对比，提示对比结果。
- 用户名和密码为空时给出提示。

完成效果

运行网页，效果如图 1.13、图 1.14 所示。

图 1.13 密码提示

图 1.14 用户名提示

技能要点

- 多重 if 结构。
- 自定义函数。

实现思路

- <input>标签中的 onclick 属性值设置为函数名，实现单击按钮对函数的调用。

```
<input type="button" value="登录" onclick="check();"/>
```

- 函数内利用多重 if 结构判断。

本章总结

- JavaScript 的基本语法以及三种调用方式。
- 条件结构 if 语句及 switch 语句的用法。
- 自定义函数使用关键字 function。
- 调用函数常使用的格式：事件名＝"函数名()" ;。

本章作业

1．根据你的理解，简述 JavaScript 脚本的执行原理。

2．简述 JavaScript 的组成以及每部分的作用。

3．简述 JavaScript 常用的基本数据类型有哪些。

4．使用 prompt()方法在页面中弹出提示对话框，根据用户输入星期一～星期日的不同，弹出不同的信息提示对话框，要求使用函数实现，具体要求如下：

- 输入“星期一”时，弹出“新的一周开始了”。
- 输入“星期二”“星期三”“星期四”时，弹出“努力工作”。
- 输入“星期五”时，弹出“明天就是周末了”。
- 输入“星期六”“星期日”时，弹出“放松地休息”。

效果如图 1.15、图 1.16、图 1.17 所示。

5．统计语文、数学、计算机三门课程的总成绩。使用 prompt()方法在页面中弹出提示对话框，提示用户输入三门课的成绩，弹出总成绩，要求使用函数实现，具体要求如下：

- 成绩必须为数字，否则给出提示信息。
- 成绩必须大于等于零，否则给出提示信息。

效果如图 1.18、图 1.19、图 1.20 所示。

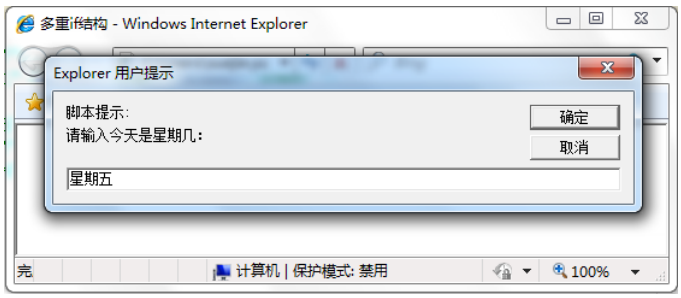

图 1.15　输入星期

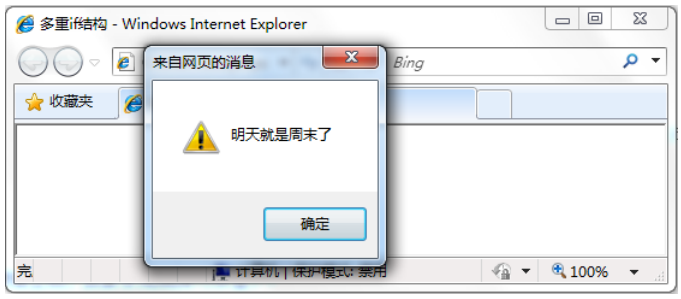

图 1.16　提示正确信息

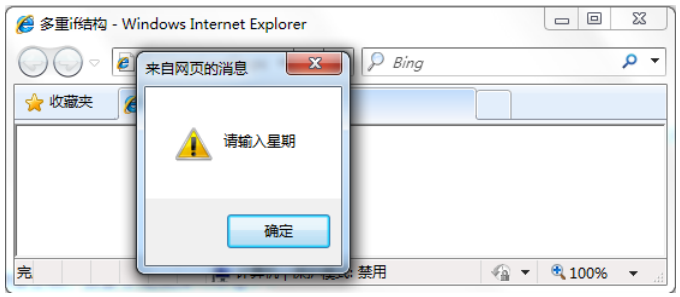

图 1.17　提示错误信息

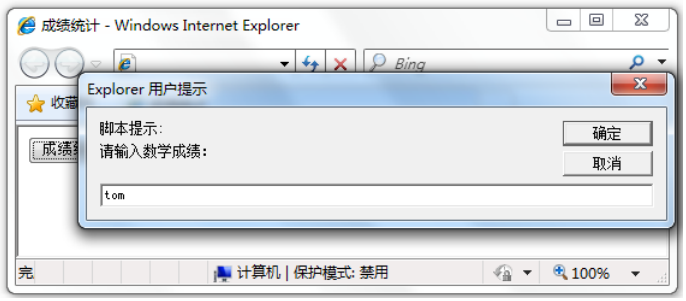

图 1.18　输入非数字成绩

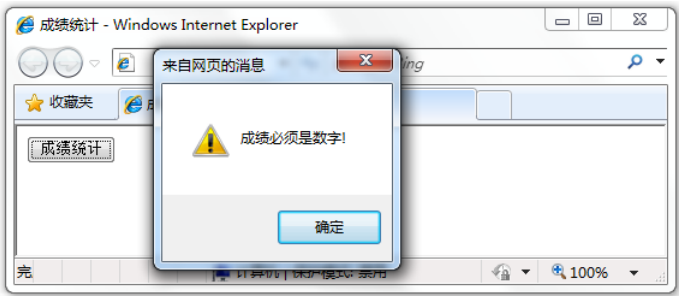

图 1.19　输入非数字成绩提示信息

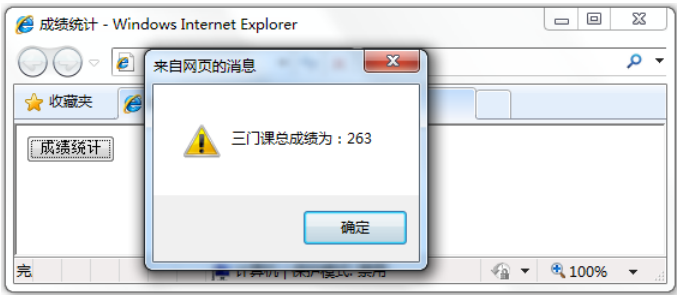

图 1.20　输入正确后统计结果

6．请登录课工场，按要求完成预习作业。