第4章

JavaScript 语言入门

4.1 项目导引——网页程序设计

在前面的 HTML、CSS 等技术的学习中，小王可以通过 HTML、CSS 可以完成一些基本的网页。小王接下来需要完成以下功能的网页时，遇到了困难。网页如图 4-1 所示，验证用户输入的身份证号和电子邮件做个简单验证，身份证验证规则为由长度 15 或者 18 位，由“0～9”和“X”组成；电子邮件验证规则为包含“@”、“.”字符。如果验证通过在弹出信息“验证通过”，如果验证不通过则提示出错的位置，并且为了增强网页效果，要求鼠标移动到网页时，按钮文字颜色改为红色，鼠标离开后恢复。

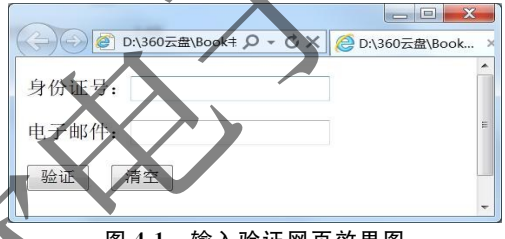

图 4-1 输入验证网页效果图

小王在实现该网页时，遇到了很大的困难：网页需要用到编程语言的功能，如函数、判断语句等，可 HTML 只是一种标识语言，主要功能是内容展示，并没有函数、过程、流控制语句等程序设计的元素。那该如何实现这些功能呢？

4.2 项目分析

从前面的学习中可以了解到，HTML 并不具备编程语言的特性，所以要实现网页编程还需要借助于其他的技术，这种技术就是脚本语言，而 JavaScript 是应用最为广泛的网页脚本语言。在下一节的技术准备中我们将具体的介绍 JavaScript 知识点。

通过本章的学习，我们将了解 JavaScript 的语法、函数、流程控制语句、事件等知识，并通过项目实例掌握如何使用 JavaScript 解决网页编程中的实际问题。

4.3 技术准备

JavaScript是学习脚本语言的首选。它兼容性好,绝大多数浏览器均支持JavaScript,而且功能强大,实现简单方便,入门简单。JavaScript是由Netscape公司创造的一种脚本语言。作为一门独立的编程语言,JavaScript可以做很多的事情,但它最主流的应用还是在Web上——创建动态网页(即网页特效)。JavaScript在网络上应用广泛,几乎所有的动态网页里都能找到它的身影。目前流行的AJAX也是依赖于JavaScript而存在的。

JavaScript与Java是由不同的公司开发的不同产品。Java是SUN Microsystems公司推出的新一代面向对象的程序设计语言,特别适合于Internet应用程序开发。

JavaScript与Jscript也不是一门相同的语言,Jscript和vbscript是微软开发的两种脚本语言,微软,Netscape公司以及其他语言开发商为减少Web开发者的兼容麻烦,所以成立ECMA,该组织专门制定脚本语言的标准和规范。ECMA制定的标准脚本语言叫做ECMAScript,JavaScript符合ECMA的标准,其实JavaScript也可以叫做ECMAScript.

JavaScript可以被嵌入到HTML文件中,直接在客户端浏览器中执行,不需要经过Web服务器就可以对用户操作作出响应,使网页更好地与用户交互;在利用客户端个人电脑性能资源的同时,适当减小服务器端的压力,并减少用户等待时间。

4.3.1 我的第一个JavaScript程序——网页中插入JavaScript

我们在VS2010中新建一网页(当然,也可以使用Dreamweaver或者记事本等工具):

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>第一个JavaScript程序</title>
</head>
<body>
    <script type="text/JavaScript">
        window.alert("hello,world!");
    </script>
</body>
</html>
```

使用浏览器打开该网页,就可以弹出一个提示框。从这个程序中,我们可以了解到,网页中插入JavaScript代码使用<script>标签。通常使用下面的代码可以在网页中插入JavaScript:

```
<script type="text/JavaScript" language="JavaScript">
    ...
</script>
```

language="JavaScript"表示使用JavaScript脚本语言,脚本语言还有vbscript、jsscript等,如果省略language属性,表示默认使用JavaScript脚本。其中的"..."就是代码的内容。

```
<script type="text/JavaScript">
    window.alert("hello,world!");
</script>
```

JavaScript 使用 window.alert()来弹出提示框，也可以省略“window.”，直接写“alert()”。上述程序将会输出在网页上弹出“hello,world!”的提示框（注意：alert()是警告提示对话框；confirm()是确认对话框；prompt()是提问对话框）。

“hello,world!”两侧双引号代表字符串的意思。在 JavaScript 中，一行的结束就被认定为语句的结束，但是最好还是要在结尾加上一个分号“;”来表示语句的结束。这是一个编程的好习惯，事实上在很多语言中句末的分号都是必须的。

网页执行效果如图 4-2 所示：

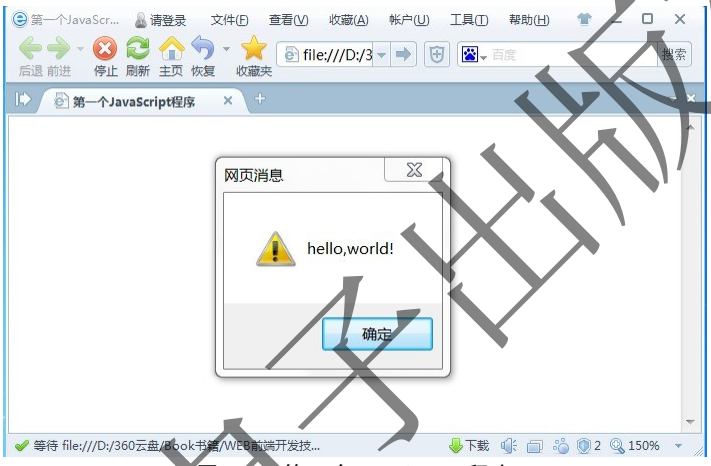

图 4-2　第一个 JavaScript 程序

JavaScript 脚本可以放在网页的 head 里或者 body 部分，而且效果也不相同。上例代码中，放在 body 部分的 JavaScript 脚本在网页读取到该语句的时候就会执行。

JavaScript 代码如果放在 head 标签中，则该部分的代码在被调用的时候才会执行，例如：

```
<html>
<head>
<script type="text/JavaScript">
....
</script>
</head>
```

通常是在<script>...</script>定义函数，通过调用函数来执行 head 里的脚本。

也可以像添加外部 CSS 一样添加外部 JavaScript 脚本文件，其后缀通常为.js。例如：

```
<html>
<head>
<script src="scripts.js"></script>
</head>
<body>
</body>
</html>
```

如果很多网页都需要包含一段相同的代码,那么将这些代码写入一个外部JavaScript文件是最好的方法。此后,任何一个需要该功能的网页,只需要引入这个js文件就可以了。

注意:脚本文件里头不能再含有<script>标签。

注:放在body里的函数是一个例外,它并不会被执行,而是等被调用时才会执行。关于函数与调用的概念将在后面讲到。

4.3.2 JavaScript基本语法

1. JavaScript注释

HTML、CSS里都有注释,JavaScript里自然也有注释,而且分为单行注释与普通注释两种。插入单行注释的符号是"//";多行注释以"/*"开始,以"*/"结束。

```
<script type="text/JavaScript">
// 我是单行注解
document.write("<p>窗前明月光,</p>");
/*
我是多行注解
可以写多行
*/
document.write("<p>疑是地上霜。</p>");
</script>
```

注释的作用就是记录自己在编程时候的思路,以便以后自己阅读代码时可以马上找到思路。同样,注释也有助于别人阅读自己书写的JavaScript代码。总之书写注释是一个良好的编程习惯。

2. JavaScript变量

(1)声明变量。

```
我们可以用 var 声明变量,比如:
<script type="text/JavaScript">
var a ; //声明一个变量 a
a=5 ; //给变量赋值
</script>
```

其实在第一个例子中我们已经看到了,JavaScript中可以不声明变量直接赋值。不过先声明变量是一个良好的编程习惯。变量命名有如下规定:

①变量名区分大小写,A与a是两个不同变量。

②变量名必须以字母或者下划线开头。

(2)给变量赋值。

JavaScript中使用"="给变量赋值:

```
a=1;// 将 1 赋值给 a
a=10;// 将 10 赋值给 a
```

(3)变量数据类型。

其实，在 JavaScript 中，变量是无所不能的容器，你可以把任何东西存储在变量里，例如：

```
varn=123;  //数字
varstr1="abcd";  //字符串
```

变量还可以存储更多的东西，例如数组、对象、布尔值等等。

3. JavaScript 操作符

(1)运算操作符。

操作符是用于在 JavaScript 中指定一定动作的符号，其中算术操作符主要用来完成类似加减乘除的工作。操作符有优先级，在数学中，“a+b * c”这个式子中，乘法将先于加法运算。同样，在 JavaScript 中，这个式子会按相同的顺序执行。我们称之为“优先级”，即“*”的优先级高于“+”。与数学中一样，改变运算顺序的方法是添加括号，JavaScript 中改变优先级的方法也是添加括号，例如：

```
(a+b) * c;
```

JavaScript 的操作运算符与 C 语言等编程语言极为类似，下面通过举例说明：

```
str="hello"+" world"; // 还可以用于字符串的连接
a=10;                 //变量赋值
a++;                  //a 的值变为 9
a--                   //a 的值有变回 10
a=a+1;                //等同于 a++
a+=1;                 //将 a 的值加 1 之后再赋给 a
a -=1;                // a=a -1
a *=2;                // a=a * 2
a /=2;                // a=a / 2
```

(2)比较操作符。

比较操作符“==”，它表示的意思就是“相等吗?”，例如：a==b 表示“a 与 b 的值相等吗?”。

在 JavaScript 中，这样的比较操作符有很多，下面就列出这些操作符以及它们的含义。

- “>”—— a 大于 b 吗?
- “<”—— a 小于 b 吗?
- “>=”—— a 大于等于 b 吗?
- “<=”—— a 小于等于 b 吗?
- “==”—— a 等于 b 吗?
- “!=”—— a 不等于 b 吗?

(3)逻辑操作符。

数学里面的“a>b”在 JavaScript 中还表示为“a>b”；数学中的“b 大于 a，b 小于 c”是“a<b<c”，那么在 JavaScript 中是不是也一样呢? 对不起，JavaScript 没有那么聪明，你需要这么写：

```
b>a && a<b
```

“&&”是而且的意思。

```
if(条件1 && 条件2)
{//代码}
```

只有条件1和2同时满足,代码才会得到执行。类似的操作符还有“或者(||)”和“非(!)”

4.3.3 JavaScript控制语句

1. if else语句(如果,否则)

if else是所有编程语言里都有的功能,它使得程序具有简单的判断能力。例如:

```
<script type="text/JavaScript">
varscore=75
if (score >=60)
{
document.write("不错,及格。");
}
else
{
document.write("补考。");
}
</script>
```

上面的代码用到了“else”,它会给if添加一种“否则”的状态。当score大于等于60分时,输出及格信息,否则输出补考信息。

如果想做更多的判断,可以用if的嵌套,看下面的代码。

```
<script type="text/JavaScript">
var score=85
if (score >=80)   // 成绩大于等于80
{
document.write("成绩很好。");
}
else if (score>=60)   //成绩小于80,但大于等于60
{
document.write("Pass");
}
else  成绩小于60
{
document.write("不及格!");
}
</script>
```

2. switch语句

当有很多种选项的时候,switch比if else使用更方便。比如要实现如下功能的程序:输入一个学生的考试成绩,我们按照每十分一个等级将成绩分等,程序将根据成绩的等级做出不同

的评价。

很明显，用 if else 可以实现这样的程序，但是代码会很复杂。而如果使用 switch 语句，代码则会简单一些，首先来看一下思路，再把它翻译成 JavaScript。

思路：

- 将分数转化为特定等级以便于 switch 处理。
- 判断分数属于哪种等级。
- 根据分数等级做出评价：例如低于 60 给出挂科评价。

翻译成 JavaScript 就是如下代码(注意注释)：

```
    <script type="text/JavaScript">
    //首先，我们用 score 变量来存储分数，假设为 65
    var score=65;
    //用分数除以 10，parseInt 的作用是把它转换为整数，
    //暂时不用深究，()内最后的结果为 6
    switch (parseInt(score / 10)) {
    //switch 开始实现判断过程，case 6 得到满足
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
    //根据不同的等级做出不同的行为。
    //冒号后面的语句就是行为
    //case0 到 5 的行为都是下面这个语句
    degree="恭喜你，又挂了!";
    break;
    case 6:
    degree="勉强及格";
break;
    case 7:
    degree="凑合，凑合"
    break;
    case 8:
    degree="不错，不错";
    break;
    case 9:
    case 10:
    degree="高手高手，佩服佩服";
    }//end of switch
    </script>
```

在每个 case 所执行的语句里添加上一个 break 语句，否则后面的语句还是会执行，就没有

意义了。

3. for 循环

for 用来实现循环执行某段代码功能，其语句结构如下：

```
for(初始条件;判断条件;循环后动作)
{
    循环代码;
}
```

让我们来看一个简单的例子吧：求 1+2+3+…+100=？

```
<script type="text/JavaScript">
var i=1;
var sum=0;
for (i=1;i<=100;i++)
{
   sum=sum+i;
}
document.write("1+2+3+…+100="+sum);
</script>
```

在上面那个例子中，循环恰好执行了 100 次，那么和"for (i=1;i<=100;i++)"一句中的 100 是不是 100 次的意思呢？下面我们就来看看 for 循环的工作机制。

首先"i=1"叫做初始条件，也就是说从哪里开始，我们的例子从 i=1 开始。

出现在第一个分号后面的"i<=100"表示判断条件，每次循环都会先判断这个条件是否满足，如果满足则继续循环，否则停止循环，继续执行 for 循环后面的代码。我们设定了 i=0，岂不是永远都小于等于 100 吗？来看第三个部分。

最后的"i++"表示让 i 在自身的基础上加 1，这时每次循环后的动作.也就是说，每次循环结束，i 都会比原来大 1，执行若干次循环之后，i<=100 的条件就不满足了，这时循环结束。for 循环后面的代码将得到执行。

4. while 循环

while 循环重复执行一段代码，直到某个条件不再满足。

(1)while 循环的结构。

```
while(判断条件)
{
        循环代码;
}
```

其实 while 循环和 for 循环的作用都是重复执行代码，例如下面这段代码，和上一节 for 循环的输出结果完全没有区别。代码如下。

```
<script type="text/JavaScript">
var i=1;
var sum=0;
while (i<=100)
{
    sum=sum+i;
    i++;
}
document.write("1+2+3+…+100="+sum);
</script>
```

(2)do while 循环的结构。

do wile 结构的基本原理和 while 结构是基本相同的,但是它保证循环体至少被执行一次。因为它是先执行代码,后判断条件。例子如下:

```
<script type="text/JavaScript">
var i=1;
var sum=0;
do
{
   sum=sum+i;
   i++;
}
while (i<=100)
document.write("1+2+3+…+100="+sum);
</script>
```

注意 Break 于 Continue 的区别,Break 可以跳出整个循环,Continue 跳过本次循环。

5. For... In 循环

JavaScript 中的 for in 循环通常用来遍历数组。

首先要了解什么是数组,所谓数组,其实就是一个保存了一组类似变量的一个集合。我们来看一个保存了水果的数组实例:

```
<script type="text/JavaScript">
var x;
var fruit=new Array();//创建一个新的数组
fruit[0]="Apple";
fruit[1]="Bananer";
fruit[2]="Cherry ";
for (x in fruit)//数组中的每一个变量
{
document.write(fruit[x]+"<br />");
}
</script>
```

输出结果如下：

```
Apple
Bananer
Cherry
```

我们来分析一下上面的例子：

“varfruit=new Array();”一句创建了一个新的数组。

“fruit[0]="Apple";”以及之后的两句则是给fruit数组赋值。这与我们之前见过的变量赋值不太一样，fruit后面多出一个“[0]”，这个是变量的索引。我们之前已经说了，数组是变量的集合，因此我们在赋值之前需要指明给数组中的哪一个变量赋值。在这里，“[0]”表示的是fruit数组所包含的第一个变量。

4.3.4 函数

1. 函数定义

函数定义格式：

```
function 函数名()
{
        函数代码;
}
```

函数由关键字function定义，把“函数名”替换为你想要的名字，把“代码”替换为完成特定功能的代码，函数就定义好了。了解了如何定义函数，我们就来自己编写一个实现两数相加的简单函数吧。

首先给函数起一个有意义的名字：“hi”，它的代码如下：

```
function hi(){
  alert("hi,你好!");
}
```

2. 函数的调用

可以通过很多种方法调用上面的函数，我们这里使用最简单的函数调用方式——按钮的点击事件，JavaScript事件会在后面介绍。试着点击下面的按钮调用定义好的函数：

```
<html>
<head>
<script language=JavaScript>
        function hi(){
  alert("hi,你好!");
}
</SCRIPT>
</head>
<body>
<form>
<input type="button" value="打招呼" ONCLICK="hi()">
</form>
</body>
</html>
```

通过 button 按钮的鼠标单击事件 onclick 调用 hi()函数。

3. 带参数的函数

上述 hi()函数不能实现任意指定两数相加。其实,函数的定义可以是下面的格式:

```
function(参数 1,参数 2,参数 3){
        部分函数代码……
……
}
```

按照这个格式,我们的函数应该写成:

```
function hi(language){
if (language==="Chinese"{
        alert("早上好!");}
else if(language==="English"{
    alert("Good Morning!");}
}
```

4. 带返回值的函数

使用 return 返回函数值,例如:

```
function area(r){
        return r*r*3.14;
}
```

return 后面的值叫做返回值。使用下面的语句调用函数就可以将这个返回值存储在变量中了。

```
result=area(10);
```

该语句执行后,result 变量中的值为 314。

4.3.5　事件

使用点击事件调用,需要给元素设置 onclick 属性。示例代码如下:

```
<button value="点击提交" onclick="displaymessage()">onclick 调用函数</button>
```

由于设置了 onclick="displaymessage()",因此点击按钮则会调用函数displaymessage()。

使用鼠标经过事件调用函数的代码如下:

```
<button value="点击提交" onmouseover="displaymessage()">鼠标滑过调用函数</button>
```

当鼠标经过按钮时,触发 onmouseover 事件,将会调用函数 displaymessage()。

使用鼠标移出事件调用函数的代码如下:

```
<button value="点击提交" onmouseout="displaymessage()">鼠标移出调用函数</button>
```

把鼠标移动到这个按钮里面,当再移动出去时,触发 onmouseout 事件,将会调用函数 displaymessage()。

JavaScript中还有很多事件，完整的列表可以看看 http://www.w3pop.com/learn/view/p/3/o/0/doc/jsref_events/。

HTML标签中事件动作的属性如表4-1所示。

表4-1 HTML事件列表

属性	事件发生时机
onabort	图片下载被打断时
onblur	元素失去焦点时
onchange	框内容改变时
onclick	鼠标点击一个对象时
ondblclick	鼠标双击一个对象时
onerror	当加载文档或图片时发生错误时
onfocus	当元素获取焦点时
onkeydown	按下键盘按键时
onkeypress	按下或按住键盘按键时
onkeyup	放开键盘按键时
onload	页面或图片加载完成时
onmousedown	鼠标被按下时
onmousemove	鼠标被移动时
onmouseout	鼠标离开元素时
onmouseover	鼠标经过元素时
onmouseup	释放鼠标按键时
onreset	重新点击鼠标按键时
onresize	当窗口或框架被重新定义尺寸时
onselect	文本被选择时
onsubmit	点击提交按钮时
onunload	用户离开页面时

4.3.6 对象化编程

JavaScript是使用“对象化编程”的，或者叫“面向对象编程”的。所谓“对象化编程”，意思是把JavaScript能涉及的范围划分成大大小小的对象，对象下面还继续划分对象直至非常详细为止，所有的编程都以对象为出发点，基于对象。小到一个变量，大到网页文档、窗口甚至屏幕，都是对象。

JavaScript对象是可以是一段文字、一幅图片、一个表单(Form)等等。每个对象有它自己的属性、方法和事件。对象的属性是反映该对象某些特定的性质的，例如：字符串的长度、图像的长宽、文字框(Textbox)里的文字等等。对象的方法能对该对象做一些事情，例如，表单的

“提交”(Submit),窗口的“滚动”(Scrolling)等等。而对象的事件就能响应发生在对象上的事情,例如提交表单产生表单的“提交事件”,点击连接产生的“点击事件”。不是所有的对象都有以上三个性质,有些没有事件,有些只有属性。引用对象的任一“性质”用“<对象名>.<性质名>”这种方法。

JavaScript 对象有:基本对象、全局对象、文档对象。

1. 基本对象

(1)String 字符串对象。

我们在之前的学习中已经就在使用字符串对象对象了,声明一个字符串对象的方法就是直接赋值。比如:

```
var s="我有个 7 个字符";
```

定义了 s 这个字符串之后,我们就有了一个字符串对象,我们可以访问它的属性,使用它的方法。

属性

length 用法:<字符串对象>.length;返回该字符串的长度。

方法

charAt()用法:<字符串对象>.charAt(<位置>);返回该字符串位于第<位置>位的单个字符。注意:字符串中的一个字符是第 0 位的,第二个才是第 1 位的,最后一个字符是第 length−1 位的。

charCodeAt()用法:<字符串对象>.charCodeAt(<位置>);返回该字符串位于第<位置>位的单个字符的 ASCII 码。

fromCharCode()用法:String.fromCharCode(a, b, c...);返回一个字符串,该字符串每个字符的 ASCII 码由 a, b, c... 等来确定。

indexOf()用法:<字符串对象>.indexOf(<另一个字符串对象>[, <起始位置>]);该方法从<字符串对象>中查找<另一个字符串对象>(如果给出<起始位置>就忽略之前的位置),如果找到了,就返回它的位置,没有找到就返回“−1”。所有的“位置”都是从零开始的。

lastIndexOf()用法:<字符串对象>.lastIndexOf(<另一个字符串对象>[, <起始位置>]);跟 indexOf() 相似,不过是从后边开始找。

split()用法:<字符串对象>.split(<分隔符字符>);返回一个数组,该数组是从<字符串对象>中分离开来的,<分隔符字符>决定了分离的地方,它本身不会包含在所返回的数组中。例如:'1&2&345&678'.split('&')返回数组:1,2,345,678。关于数组,我们等一下就讨论。

substring()用法:<字符串对象>.substring(<始>[, <终>]);返回原字符串的子字符串,该字符串是原字符串从<始>位置到<终>位置的前一位置的一段。<终> −<始>=返回字符串的长度(length)。如果没有指定<终>或指定得超过字符串长度,则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串,则返回空字符串。

substr()用法:<字符串对象>.substr(<始>[, <长>]);返回原字符串的子字符串,该字符串是原字符串从<始>位置开始,长度为<长>的一段。如果没有指定<长>或指定得超过字符串长度,则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串,则返回空字符串。

toLowerCase()用法:<字符串对象>.toLowerCase();返回把原字符串所有大写字母都变成小写的字符串。

toUpperCase()用法:<字符串对象>.toUpperCase();返回把原字符串所有小写字母都变成大写的字符串。

(2)Array 数组对象。

数组对象是一个对象的集合,里边的对象可以是不同类型的。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,是从零开始的。

数组的定义方法:

var <数组名>=new Array();

这样就定义了一个空数组。以后要添加数组元素,就用:

<数组名>[<下标>]=...;

数组的下标用方括号括起来。

如果想在定义数组的时候直接初始化数据,请用:

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

例如:

var myArray=new Array(1, 4.5, 'Hi');

定义了一个数组 myArray,里边的元素是:myArray[0]=1; myArray[1]=4.5; myArray[2]='Hi'。

定义时指定有 n 个空元素的数组,请用:

var a=new Array(n);

属性

length 用法:<数组对象>.length;返回:数组的长度,即数组里有多少个元素。它等于数组里最后一个元素的下标加一。

方法

join()用法:<数组对象>.join(<分隔符>);返回一个字符串,该字符串把数组中的各个元素串起来,用<分隔符>置于元素与元素之间。这个方法不影响数组原本的内容。

reverse()用法:<数组对象>.reverse();使数组中的元素顺序反过来。如果对数组[1, 2, 3]使用这个方法,它将使数组变成:[3, 2, 1]。

slice()用法:<数组对象>.slice(<始>[, <终>]);返回一个数组,该数组是原数组的子集,始于<始>,终于<终>。如果不给出<终>,则子集一直取到原数组的结尾。

sort()用法:<数组对象>.sort([<方法函数>]);使数组中的元素按照一定的顺序排列。如果不指定<方法函数>,则按字母顺序排列。在这种情况下,80 是比 9 排得前的。如果指定<方法函数>,则按<方法函数>所指定的排序方法排序。<方法函数>比较难讲述,这里只将一些有用的<方法函数>介绍给大家。

按升序排列数字:

```
function sortMethod(a,b) {
return a - b;
}
myArray.sort(sortMethod);
```

按降序排列数字：把上面的“a－b”该成“b－a”。

(3)Math “数学”对象。

Math 对象，提供对数据的数学计算。下面所提到的属性和方法，不再详细说明“用法”，大家在使用的时候记住用“Math.＜名＞”这种格式。

属性

E 返回常数 e (2.718281828...)。

LN2 返回 2 的自然对数 (ln 2)。

LN10 返回 10 的自然对数 (ln 10)。

LOG2E 返回以 2 为低的 e 的对数 (log2e)。

LOG10E 返回以 10 为低的 e 的对数 (log10e)。

PI 返回 π(3.1415926535...)。

SQRT1_2 返回 1/2 的平方根。

SQRT2 返回 2 的平方根。

方法

abs(x)返回 x 的绝对值。

acos(x)返回 x 的反余弦值(余弦值等于 x 的角度)，用弧度表示。

asin(x)返回 x 的反正弦值。

atan(x)返回 x 的反正切值。

atan2(x, y)返回复平面内点(x, y)对应的复数的幅角，用弧度表示，其值在-π 到 π 之间。

ceil(x)返回大于等于 x 的最小整数。

cos(x)返回 x 的余弦。

exp(x)返回 e 的 x 次幂 (e^x)。

floor(x)返回小于等于 x 的最大整数。

log(x)返回 x 的自然对数 (lnx)。

max(a, b)返回 a, b 中较大的数。

min(a, b)返回 a, b 中较小的数。

pow(n, m)返回 n 的 m 次幂 (n^m)。

random()返回大于 0 小于 1 的一个随机数。

round(x)返回 x 四舍五入后的值。

sin(x)返回 x 的正弦。

sqrt(x)返回 x 的平方根。

tan(x)返回 x 的正切。

(4)Date 对象。

Date 日期对象。这个对象可以储存任意一个日期，从 0001 年到 9999 年，并且可以精确到

毫秒数(1/1000 秒)。

定义一个日期对象:

```
var today=new Date();
```

这个方法使 d 成为日期对象,并且已有初始值:当前时间。如果要自定初始值,可以用下列方法:

```
var d=new Date(99, 10, 1);  //99 年 10 月 1 日
var d=new Date('Oct 1, 1999'); //99 年 10 月 1 日
```

最好的方法就是用下面介绍的"方法"来严格的定义时间。

方法

以下有很多 getXXX()、setXXX()这样的方法,getXXX()是获得某个数值,而 setXXX()是设定某个数值。

如无说明,方法的使用格式为:"<对象>.<方法>",下同。

get/setFullYear()返回/设置年份,用四位数表示。如果使用"x.setFullYear(99)",则年份被设定为 0099 年。

get/setYear()返回/设置年份,用两位数表示。设定的时候浏览器自动加上"19"开头,故使用"x.setYear(00)"把年份设定为 1900 年。

get/setMonth()返回/设置月份,0 表示 1 月。

get/setDate()返回/设置日期。

get/setDay()返回/设置星期,0 表示星期天。

get/setHours()返回/设置小时数,24 小时制。

get/setMinutes()返回/设置分钟数。

get/setSeconds()返回/设置秒钟数。

get/setMilliseconds()返回/设置毫秒数。

get/setTime()返回/设置时间,该时间就是日期对象的内部处理方法:从 1970 年 1 月 1 日零时正开始计算到日期对象所指的日期的毫秒数。如果要使某日期对象所指的时间推迟 1 小时,就用:"x.setTime(x.getTime()+60 * 60 * 1000);"(一小时 60 分,一分 60 秒,一秒 1000 毫秒)。

getTimezoneOffset() 返回日期对象采用的时区与格林威治时间所差的分钟数。在格林威治东方的市区,该值为负,例如:中国时区(GMT+0800)返回"-480"。

toString()返回一个字符串,描述日期对象所指的日期。这个字符串的格式类似于:"Fri Jul 21 15:43:46 UTC+0800 2000"。

toLocaleString()返回一个字符串,描述日期对象所指的日期,用本地时间表示格式。如:"2000-07-21 15:43:46"。

toGMTString()返回一个字符串,描述日期对象所指的日期,用 GMT 格式。

toUTCString()返回一个字符串,描述日期对象所指的日期,用 UTC 格式。

parse()用法:Date.parse(<日期对象>);返回该日期对象的内部表达方式。

下面例子显示当前日期:

```
<html>
<body>
<script language="JavaScript">
today=new Date();
var day; var date;
if(today.getDay()==0) day="星期日";
if(today.getDay()==1) day="星期一";
if(today.getDay()==2) day="星期二";
if(today.getDay()==3) day="星期三";
if(today.getDay()==4) day="星期四";
if(today.getDay()==5) day="星期五";
if(today.getDay()==6) day="星期六";
   date="今天是"+(today.getYear())+"年"+(today.getMonth()+1)+"月"+today.getDate()
+"日"+day+"";
document.write(date);
</script>
</body>
</html>
```

2. 全局对象

全局对象就是一些全局函数，他们可以直接用，这里对它们进行简要的介绍。

eval()把括号内的字符串当作标准语句或表达式来运行。

```
b="2+5*2";
var a=eval(b);
```

a 的值为 12。

isNaN()如果括号内的值是“NaN(不是数字)”，则返回 true 否则返回 false。

parseInt()返回把括号内的内容转换成整数之后的值。如果括号内是字符串，则字符串开头的数字部分被转换成整数，如果以字母开头，则返回“NaN”。

parseFloat()返回把括号内的字符串转换成浮点数之后的值，字符串开头的数字部分被转换成浮点数，如果以字母开头，则返回“NaN”。

toString()用法：<对象>.toString()；把对象转换成字符串。如果在括号中指定一个数值，则转换过程中所有数值转换成特定进制。

escape()返回括号中的字符串经过编码后的新字符串。该编码应用于 URL，也就是把空格写成“%20”这种格式。“+”不被编码，如果要“+”也被编码，请用：escape('...', 1)。

unescape()是 escape() 的反过程。解编括号中字符串成为一般字符串。

3. 文档对象

文档对象是指在网页文档里划分出来的对象。在 JavaScript 能够涉及的范围内有如下几个“大”对象：window，document，location，navigator，screen，history 等。下面是一个文档对象树。

要引用某个对象，就要把父级的对象都列出来。例如，要引用某表单“MyForm”的某文字

框“UserName”,就要用“document. MyForm. UserName”。

引用Form下的表单元素对象不使用名称,而是通过对象的ID或Name进行引用,或使用它所属的对象数组。比如:

```
<input id="UserName" type="text" />
…
varname=document.getElementById("UserName");//通过id获取值
```

文档对象列表如表4-2所示。

表4-2 文档对象列表

● navigator	浏览器对象
● screen	屏幕对象
● window	窗口对象
◆ history	历史对象
◆ location	地址对象
◆ frames[]; Frame	框架对象
◆ document	文档对象
■ anchors[]; links[]; Link	连接对象
■ applets[]	Java小程序对象
■ embeds[]	插件对象
■ forms[]; Form	表单对象
◇ Button	按钮对象
◇ Checkbox	复选框对象
◇ elements[]; Element	表单元素对象
◇ Hidden	隐藏对象
◇ Password	密码输入区对象
◇ Radio	单选域对象
◇ Reset	重置按钮对象
◇ Select	选择区(下拉菜单、列表)对象
◇ options[]; Option	选择项对象
◇ Submit	提交按钮对象
◇ Text	文本框对象
◇ Textarea	多行文本输入区对象
◇ images[]; Image	图片对象

(1)navigator。

navigator浏览器对象,反映了当前使用的浏览器的资料。

属性

appCodeName返回浏览器的“码名”,流行的IE和NN都返回“Mozilla”。

appName返回浏览器名,IE返回“Microsoft Internet Explorer”,NN返回“Netscape”。

appVersion返回浏览器版本,包括了大版本号、小版本号、语言、操作平台等信息。

platform返回浏览器的操作平台,对于Windows 9x上的浏览器,返回“Win32”(大小写可

能有差异)。

userAgent 返回以上全部信息。例如,IE5.01 返回“Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)”。

javaEnabled()返回一个布尔值,代表当前浏览器允许不允许 Java。

(2)screen。

screen 屏幕对象,反映了当前用户的屏幕设置。

属性

width 返回屏幕的宽度(像素数)。

height 返回屏幕的高度。

availWidth 返回屏幕的可用宽度(除去了一些不自动隐藏的类似任务栏的东西所占用的宽度)。

availHeight 返回屏幕的可用高度。

colorDepth 返回当前颜色设置所用的位数,-1:黑白,8:256 色,16:增强色,24/32:真彩色。

(3)window。

window 窗口对象是最大的对象,它描述的是一个浏览器窗口。一般要引用它的属性和方法时,不需要用“window.xxx”这种形式,而直接使用“xxx”。一个框架页面也是一个窗口。

属性

name 窗口的名称,由打开它的连接(<a target="...">)或框架页(<frame name="...">)或某一个窗口调用的 open()方法决定。一般我们不会用这个属性。

status 指窗口下方的“状态栏”所显示的内容。通过对 status 赋值,可以改变状态栏的显示。

opener 的用法:window.opener;返回打开本窗口的窗口对象。注意:返回的是一个窗口对象。如果窗口不是由其他窗口打开的,在 Netscape 中这个属性返回 null,在 IE 中返回“未定义(undefined)”。undefined 在一定程度上等于 null。注意:undefined 不是 JavaScript 常数,如果你企图使用“undefined”,那就真的返回“未定义”了。

self 指窗口本身,它返回的对象跟 window 对象是一模一样的。最常用的是“self.close()”,放在<a>标记中:“关闭窗口”。

parent 返回窗口所属的框架页对象。

top 返回占据整个浏览器窗口的最顶端的框架页对象。

history 历史对象,见下表 4-3。

location 地址对象,见下表 4-3。

document 文档对象,见下表 4-3。

表 4-3

参数	
top=#	窗口顶部离开屏幕顶部的像素数
left=#	窗口左端离开屏幕左端的像素数
width=#	窗口的宽度
height=#	窗口的高度
menubar=...	窗口有没有菜单,取值 yes 或 no
toolbar=...	窗口有没有工具条,取值 yes 或 no
location=...	窗口有没有地址栏,取值 yes 或 no
directories=...	窗口有没有连接区,取值 yes 或 no
scrollbars=...	窗口有没有滚动条,取值 yes 或 no
status=...	窗口有没有状态栏,取值 yes 或 no
resizable=...	窗口给不给调整大小,取值 yes 或 no
fullscreen=	窗口是否全屏,取值 yes 或 no

方法

● open()打开一个窗口。

用法:open(<URL字符串>, <窗口名称字符串>, <参数字符串>);

<URL字符串>:描述所打开的窗口打开哪一个网页。如果留空(''),则不打开任何网页。

<窗口名称字符串>:描述被打开的窗口的名称(window.name),可以使用'_top'、'_blank'等内建名称。这里的名称跟"<a href="..." target="...">"里的"target"属性是一样的。

<参数字符串>:描述被打开的窗口的样式。如果只需要打开一个普通窗口,该字符串留空(''),如果要指定样式,就在字符串里写上一到多个参数,参数之间用逗号隔开。

例:打开一个 400×100 的干净的窗口:

```
open('','_blank','width=400,height=100,menubar=no,toolbar=no,
location=no,directories=no,status=no,scrollbars=yes,resizable=yes')
```

open()方法有返回值,返回的就是它打开的窗口对象,因此:

var newWindow=open('','_blank');

这样把一个新窗口赋值到"newWindow"变量中,以后通过"newWindow"变量就可以控制窗口了。

● close()关闭一个已打开的窗口。

用法:window.close() 或 self.close():关闭本窗口。

<窗口对象>.close():关闭指定的窗口。

如果该窗口有状态栏,调用该方法后浏览器会警告:"网页正在试图关闭窗口,是否关闭?"然后等待用户选择是否;如果没有状态栏,调用该方法将直接关闭窗口。

● blur()使焦点从窗口移走,窗口变为"非活动窗口"。

● focus()是窗口获得焦点,变为"活动窗口"。

● scrollTo()用法:[<窗口对象>.]scrollTo(x, y);使窗口滚动,使文档从左上角数起的

(x, y)点滚动到窗口的左上角。

● scrollBy()用法:[<窗口对象>.]scrollBy(deltaX, deltaY);使窗口向右滚动 deltaX 像素,向下滚动 deltaY 像素。如果取负值,则向相反的方向滚动。

● resizeTo()用法:[<窗口对象>.]resizeTo(width, height);使窗口调整大小到宽 width 像素,高 height 像素。

● resizeBy()用法:[<窗口对象>.]resizeBy(deltaWidth, deltaHeight);使窗口调整大小,宽增大 deltaWidth 像素,高增大 deltaHeight 像素。如果取负值,则减少。

● alert()用法:alert(<字符串>);弹出一个只包含“确定”按钮的对话框,显示<字符串>的内容,整个文档的读取、Script 的运行都会暂停,直到用户按下“确定”。

● confirm()用法:confirm(<字符串>);弹出一个包含“确定”和“取消”按钮的对话框,显示<字符串>的内容,要求用户做出选择,整个文档的读取、Script 的运行都会暂停。如果用户按下“确定”,则返回 true 值,如果按下“取消”,则返回 false 值。

● prompt()用法:prompt(<字符串>[, <初始值>]);弹出一个包含“确认”“取消”和一个文本框的对话框,显示<字符串>的内容,要求用户在文本框输入一些数据,整个文档的读取、Script 的运行都会暂停。如果用户按下“确认”,则返回文本框里已有的内容,如果用户按下“取消”,则返回 null 值。如果指定<初始值>,则文本框里会有默认值。

● setTimeout()和 setInterval()的使用

这两个方法都可以用来实现在一个固定时间段之后去执行 JavaScript。不过两者各有各的应用场景。

实际上,setTimeout 和 setInterval 的语法相同。它们都有两个参数,一个是将要执行的代码字符串,还有一个是以毫秒为单位的时间间隔,当过了那个时间段之后就将执行那段代码。

不过这两个函数还是有区别的,setInterval 在执行完一次代码之后,经过了那个固定的时间间隔,它还会自动重复执行代码,而 setTimeout 只执行一次那段代码。

虽然表面上看来 setTimeout 只能应用在 on-off 方式的动作上,不过可以通过创建一个函数循环重复调用 setTimeout,以实现重复的操作:

```
showTime();
function showTime()
{
    var today=new Date();
    alert("The time is: "+today.toString());
    setTimeout("showTime()", 5000);
}
```

一旦调用了这个函数,那么就会每隔 5 秒钟就显示一次时间。如果使用 setInterval,则相应的代码如下所示:

```
setInterval("showTime()", 5000);
function showTime()
{
    var today=new Date();
    alert("The time is: "+today.toString());
}
```

这两种方法可能看起来非常像，而且显示的结果也会很相似，不过两者的最大区别就是，setTimeout方法不会每隔5秒钟就执行一次showTime函数，它是在每次调用setTimeout后过5秒钟再去执行showTime函数。这意味着如果showTime函数的主体部分需要2秒钟执行完，那么整个函数则要每7秒钟才执行一次。而setInterval却没有被自己所调用的函数所束缚，它只是简单地每隔一定时间就重复执行一次那个函数。

如果要求在每隔一个固定的时间间隔后就精确地执行某动作，那么最好使用setInterval，而如果不想由于连续调用产生互相干扰的问题，尤其是每次函数的调用需要繁重的计算以及很长的处理时间，那么最好使用setTimeout。

用setInterval命令来创建的对象，可以用clearInterval()命令来终止。比如：

```
var MyMar=setInterval(showTime(),speed);
clearInterval(MyMar);
```

(4)history。

history历史对象指浏览器的浏览历史。

属性

length历史的项数。JavaScript所能管到的历史被限制在用浏览器的“前进”“后退”键可以去到的范围。本属性返回的是“前进”和“后退”两个按键之下包含的地址数的和。

方法

back()后退，跟按下“后退”键是等效的。

forward()前进，跟按下“前进”键是等效的。

go()用法：history.go(x)；在历史的范围内去到指定的一个地址。如果x＜0，则后退x个地址，如果x＞0，则前进x个地址，如果x＝＝0，则刷新现在打开的网页。history.go(0)跟location.reload()是等效的。

(5)location。

location地址对象描述的是某一个窗口对象所打开的地址。要表示当前窗口的地址，只需要使用“location”就行了；若要表示某一个窗口的地址，就使用“＜窗口对象＞.location”。

注意：属于不同协议或不同主机的两个地址之间不能互相引用对方的location对象，这是出于安全性的需要。例如，当前窗口打开的是“www.a.com”下面的某一页，另外一个窗口(对象名为：bWindow)打开的是“www.b.com”的网页。如果在当前窗口使用“bWindow.location”，就会出错：“没有权限”。这个错误是不能用错误处理程序(Event Handler，参阅onerror事件)来接收处理的。

属性

protocol返回地址的协议，取值为'http:','https:','file:'等等。

hostname返回地址的主机名，例如，一个“http://www.microsoft.com/china/”的地址，location.hostname＝＝'www.microsoft.com'。

port返回地址的端口号，一般http的端口号是'80'。

host返回主机名和端口号，如：'www.a.com:8080'。

pathname返回路径名，如“http://www.a.com/b/c.html”，location.pathname＝＝'b/c.html'。

hash 返回“#”以及以后的内容，如“http://www.a.com/b/c.html#chapter4”，location.hash==='#chapter4'；如果地址里没有“#”，则返回空字符串。

search 返回“?”以及以后的内容，如“http://www.a.com/b/c.asp?selection=3&jumpto=4”，location.search==='?selection=3&jumpto=4'；如果地址里没有“?”，则返回空字符串。

href 返回以上全部内容，也就是说，返回整个地址。在浏览器的地址栏上怎么显示它就怎么返回。如果想一个窗口对象打开某地址，可以使用“location.href='...'”，也可以直接用“location='...'”来达到此目的。

方法

reload()相当于按浏览器上的“刷新”(IE)或“Reload”(Netscape)键。

replace()打开一个 URL，并取代历史对象中当前位置的地址。用这个方法打开一个 URL 后，按下浏览器的“后退”键将不能返回到刚才的页面。

(6)document。

document 文档对象 描述当前窗口或指定窗口对象的文档。它包含了文档从<head>到</body>的内容。

用法：document(当前窗口)或 <窗口对象>.document (指定窗口)

属性

lastModified 当前文档的最后修改日期，是一个 Date 对象。

referrer 如果当前文档是通过点击连接打开的，则 referrer 返回原来的 URL。

title 指<head>标记里用<title>...</title>定义的文字。在 Netscape 里本属性不接受赋值。

fgColor 指<body>标记的 text 属性所表示的文本颜色。

bgColor 指<body>标记的 bgcolor 属性所表示的背景颜色。

linkColor 指<body>标记的 link 属性所表示的连接颜色。

alinkColor 指<body>标记的 alink 属性所表示的活动连接颜色。

vlinkColor 指<body>标记的 vlink 属性所表示的已访问连接颜色。

方法

open()打开文档以便 JavaScript 能向文档的当前位置(指插入 JavaScript 的位置)写入数据。通常不需要用这个方法，在需要的时候 JavaScript 自动调用。

write()和 writeln()都是向文档写入数据，所写入的会当成标准文档 HTML 来处理。writeln() 与 write() 的不同点在于，writeln() 在写入数据以后会加一个换行。这个换行只是在 HTML 中换行，具体情况能不能够是显示出来的文字换行，要看插入 JavaScript 的位置而定。如在<pre>标记中插入，这个换行也会体现在文档中。

clear()清空当前文档。

close()关闭文档，停止写入数据。如果用了 write[ln]() 或 clear() 方法，就一定要用 close() 方法来保证所做的更改能够显示出来。如果文档还没有完全读取，也就是说，JavaScript 是插在文档中的，那就不必使用该方法。

(7)anchors[]、links[]、Link。

anchors[]、links[]、Link的连接对象。

用法:document.anchors[[x]]; document.links[[x]]; <anchorId>; <linkId>

document.anchors是一个数组,包含了文档中所有锚标记(包含name属性的<a>标记),按照在文档中的次序,从0开始给每个锚标记定义了一个下标。

document.links也是一个数组,包含了文档中所有连接标记(包含href属性的<a>标记和<map>标记段里的<area>标记),按照在文档中的次序,从0开始给每个连接标记定义了一个下标。

如果一个<a>标记既有name属性,又有href属性,则它既是一个Anchor对象,又是一个Link对象。

在IE中,如果在<a>标记中添加"id="..""属性,则这个<a>对象被赋予一个标识(ID),调用这个对象的时候只需要使用"<id>"就行了。很多文档部件都可以用这个方法来赋予ID,但要注意不能有两个ID相同。

anchors和links作为数组,有数组的属性和方法。单个Anchor对象没有属性,单个Link对象的属性见下。

属性

protocol、hostname、port、host、pathname、hash、search、href与location对象相同。

target返回/指定连接的目标窗口(字符串),与<a>标记里的target属性是一样的。

4.4 项目实施

学习了前面的知识点后,我们就可以解决本章导引部分提出的问题。

【细化分析】

步骤一,设计网页界面。

参考前面图4-1,可以使用VS2008或者Dreamweaver来设计页面,注意对其中的控件命名如下:身份证输入框:txtId ; Email输入框:txtEmail;验证按钮:btnCheck;清空按钮:btnClear。

步骤二,定义身份证和Email的验证函数。

根据要求的验证规则定义身份证验证函数IdValid(id)和Email验证函数EmailValid(email),这里会使用到较多的字符串函数,请参考后面的代码和注解。然后定义Check函数,通过调用上面两个函数,实现对身份证和Email的验证。

说明:最准确的身份证验证和Email是使用正则表达式,这里仅仅根据已有规则做一定限度的验证。

步骤三,定义修改按钮文字颜色的函数。

根据要求,定义改变按钮文字颜色的函数SetColor()和恢复颜色的函数ResetColor(),注意传入当前按钮作为参数,请参考后面的代码和注解。

步骤四,在相应事件中调用函数。

在对应按钮onclic,onmouseove,onmouseout事件中指定要执行的函数。

【代码实现】

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title></title>
<script type="text/JavaScript">
    {
        //身份证验证函数
        function IdValid(id) {
            var i, temp;
            var pass=true;
            strid=id.trim();
            //是否是否为 15 或者 18
            if (strid.length==null || (strid.length != 15 && strid.length != 18))
                return false;
            for (i=0; i < strid.length; i++) {
                temp=strid.substring(i, i+1);
                //是否有 0-9 和 X 组成
                if (! ((temp >="0" && temp <="9") || temp=="x" || temp=="X")) {
                    pass=false;
                    break;
                }
            }
            return pass;
        }
        // email 验证函数
        function EmailValid(email) {
            pos1=email.indexOf("@");
            pos2=email.indexOf(".");
            len=email.length;
            // 判断包含"@"和"."字符,而且这两个字符不在最后
            if ((pos1 <=0) || (pos1==len) || (pos2 <=0) || (pos2==len) || (pos1 >
        pos2))
                return false;
            else return true;
        }
        // 验证身份证和 Email
        function Check() {
            var id=txtId.value;
            if (! IdValid(id)) {
```

```
                alert("请输入有效的身份证号!");
                // 将光标定位到身份证输入框
                txtId.focus();
                return;
            }
            var mail=txtEmail.value;
            if (! EmailValid(mail)) {
                alert("请输入有效的 Email 地址!");
                // 将光标定位到 Email 输入框
                txtEmail.focus();
                return;
            }
            alert("验证通过!");
        }
        //清空输入框
        function Clear() {
            txtEmail.value="";
            txtId.value="";
        }
        //设置按钮文字颜色
        function SetColor(btn) {
            btn.style.color="red";
        }
        //恢复按钮文字颜色
        function ResetColor(btn) {
            btn.style.color="";
        }
    }
</script>
</head>
<body>
    <p>身份证号:<input id="txtId" type="text" /></p>
    <p>电子邮件:<input id="txtEmail" type="text" /></p>
    <p><input id="btnCheck" type="button" value="验证"
    onclick="Check()" onmouseover="SetColor(this)" onmouseout="ResetColor(this)" />
    <input id="btnClear" type="button" value="清空"
    onclick="Clear()" onmouseover="SetColor(this)" onmouseout="ResetColor(this)" /></
p>
</body>
</html>
```

【运行结果】

运行结果如图 4-3，当输入不符合规则的身份证号或者 Email 时，弹出提示框。

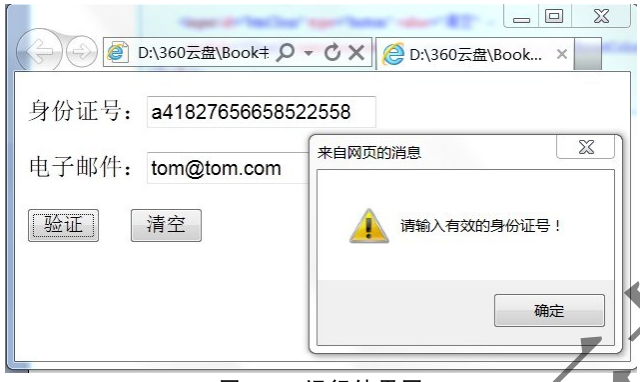

图 4-3　运行结果图

4.5　技术拓展

4.5.1　JavaScript 编码规范

1. JavaScript 文件引用

JavaScript 程序应该尽量放在 .js 的文件中，需要调用的时候在 HTML 中以 ＜script src＝"filename.js"＞ 的形式包含进来。JavaScript 代码若不是该 HTML 文件所专用的，则应尽量避免在 HTML 文件中直接编写 JavaScript 代码。因为这样会大大增加 HTML 文件的大小，无益于代码的压缩和缓存的使用。

另外＜script src＝"filename.js"＞ 标签应尽量放在文件的后面。这样会降低因加载 JavaScript 代码而影响页面中其它组件的加载时间。

2. 代码排版

每行代码应小于 80 个字符。如果代码较长，应尽量选择换行，下一行代码应缩进 4 个空格。这样可以使代码排版整齐，减轻阅读代码的疲劳感。

JavaScript 语句应该以分号结束。但大多数浏览器允许不写分号，只要在本应是分号的地方有一个换行符就行。但是如果代码行较长需要换行的时候，有哪些注意事项呢？换行应选择在操作符和标点符号之后，最好是在逗号“,”之后，而不要在变量名、字符串、数字、或“)”“]”“＋＋”“--”等符号之后换行。

3. 注释

```
＜script language＝"JavaScript"＞
//以下语句对全局变量进行初始化              (好的注释)
var valueA＝0;     //初始化 valueA 为 0          (无意义的注释)
var valueB＝1;
...
＜/script＞
```

这样的注释方式在JavaScript代码中经常见到。“初始化valueA为0”这样的注释有什么用呢？难道阅读程序的工程师从“var valueA=0;”语句中看不出来么？

此外，JavaScript的注释有两种“//”和“/* .... */”，建议“//”用作代码行注释，“/* .... */”形式用作对整个代码段的注销，或较正式的声明中，如函数参数、功能、文件功能等的描述中。

4. 标识符命名

JavaScript中的标识符的命名规则：

- 以字母、下划线“_”或美元符号“$”开头
- 允许名称中包含字母，数字，下划线“_”和美元符号“$”
- 区分大小写

变量、参数、成员变量、函数等名称均以小写字母开头，构造器的名称以大写字母开头。下划线“_”开头的变量一般习惯于标识私有或局部成员。而美元符号“$”开头的变量习惯于标识系统相关，比如系统进程等。应避免用下划线“_”或美元符号“$”来命名标识符。尽可能地降低代码的阅读负担。

5. 变量的声明

尽管JavaScript语言并不要求在变量使用前先对变量进行声明。但我们还是应该养成这个好习惯。这样可以比较容易的检测出那些未经声明的变量，避免其变为隐藏的全局变量，造成隐患。

在函数的开始应先用var关键字声明函数中要使用的局部变量，注释变量的功能及代表的含义，且应以字母顺序排序。每个变量单独占一行，以便添加注释。这是因为JavaScript中只有函数的{}表明作用域，用var关键字声明的局部变量只在函数内有效，而未经var声明的变量则被视为全局变量。

最好把每个变量的声明语句单独放到一行，并加上注释说明。所有变量按照字母排序。

```
var currentEntry; // 当前选择项
```

JavaScript没有块范围，所以在块里面定义变量很容易引起C/C++/Java程序员们的误解。在函数的首部定义所有的变量。

不要把“_”(下划线)作为变量名的第一个字符。它有时用来表示私有变量，但实际上JavaScript并没提供私有变量的功能。如果私有变量很重要，那么使用私有成员的形式。应避免使用这种容易让人误解的命名习惯。

全局变量应该全部大写。

4.5.2 JavaScript与正则表达式

在编写处理字符串的程序或网页时，经常会有查找符合某些复杂规则的字符串的需要。正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。要实现更复杂的输入验证，可以使用正则表达式。

正则表达式功能强大而复杂，在此通过一个例子讲解正则表达式的入门，和列举一些常用的正则表达式。

1. 正则表达式入门

假设你在一篇文章里里查找“hi”，你可以使用正则表达式“hi”。

这几乎是最简单的正则表达式了，它可以精确匹配这样的字符串：由两个字符组成，前一个字符是 h，后一个是 i。通常，处理正则表达式的工具会提供一个忽略大小写的选项，如果选中了这个选项，它可以匹配 hi，HI，Hi，hI 这四种情况中的任意一种。

不幸的是，很多单词里包含 hi 这两个连续的字符，比如 him，history，high 等等。用 hi 来查找的话，这里边的 hi 也会被找出来。如果要精确地查找 hi 这个单词的话，我们应该使用“\bhi\b”。

“\b”是正则表达式规定的一个特殊代码（也称为元字符，metacharacter），代表着单词的开头或结尾，也就是单词的分界处。虽然通常英文的单词是由空格，标点符号或者换行来分隔的，但是“\b”并不匹配这些单词分隔字符中的任何一个，它只匹配一个位置。

如果需要更精确的说法，“\b”匹配这样的位置：它的前一个字符和后一个字符不全是（一个是，一个不是或不存在）“\w”。

假如你要找的是“hi”后面不远处跟着一个 Lucy，你应该用“\bhi\b. * \bLucy\b”。

这里，是另一个元字符，匹配除了换行符以外的任意字符。“ * ”同样是元字符，不过它代表的不是字符，也不是位置，而是数量——它指定“ * ”前边的内容可以连续重复使用任意次以使整个表达式得到匹配。因此，“ * ”连在一起就意味着任意数量的不包含换行的字符。现在“\bhi\b. * \bLucy\b”的意思就很明显了：先是一个单词 hi，然后是任意个任意字符（但不能是换行），最后是 Lucy 这个单词。

换行符就是‘\n’，ASCII 编码为 10（十六进制 0x0A）的字符。

如果同时使用其它元字符，我们就能构造出功能更强大的正则表达式。比如“0\d\d-\d\d\d\d\d\d\d\d”，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（也就是中国的电话号码。当然，这个例子只能匹配区号为 3 位的情形）。

这里的“\d”是个新的元字符，匹配一位数字（0，或 1，或 2，或……）。“-”不是元字符，只匹配它本身——连字符（或者减号，或者中横线，或者随你怎么称呼它）。

为了避免那么多烦人的重复，我们也可以这样写这个表达式：0\d{2}-\d{8}。这里“\d”后面的“{2}”“({8})”的意思是前面“\d”必须连续重复匹配 2 次（8 次）。

2. 常用正则表达式

常见的正则表达式例子：

(1)匹配中文字符的正则表达式：[\u4e00-\u9fa5]

(2)匹配双字节字符（包括汉字在内）：[^\x00-\xff]

(3)匹配空行的正则表达式：\n[\s|]*\r

(4)匹配 HTML 标记的正则表达式：/<(.*)>.*<\/\1>|<(.*) \/>/

(5)匹配首尾空格的正则表达式：(^\s*)|(\s*$)

(6)匹配 Email 地址的正则表达式：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

(7)匹配网址 URL 的正则表达式：http://([\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)?

下面是应用正则表达式检测的函数。

应用扩展判断是否是数值：

```
function regIsNumber(fData)
{
  var reg=new RegExp("^[-]? [0-9]+[\.]? [0-9]+ $ ");
  return reg.test(fData)
}
```

验证 Email 是否正确：

```
function regIsEmail(fData)
{
  var reg=new RegExp("-[0-9a-zA-Z]+@[0-9a-zA-Z]+[\.]{1}[0-9a-zA-Z]+[\.]? [0-9a-zA-Z]+ $ ");
  return reg.test(fData);
}
```

判断手机号是否正确

```
function regIsPhone(fData)
{
   var reg=/-(\+86)? (1[0-9]{10}) $ /;
  return reg.test(fData);
}
```

4.6 本章小结

本章讲解了一个项目——一个简单的网页输入验证程序。首先提出项目需求，接下来对项目进行分析。然后就项目要用到的 JavaScript 技术进行介绍，在介绍完这些技术基础之后，一步一步的讲解如何实现这个项目。

完成项目后，对 JavaScript 的编码规范，JavaScript 与正则表达式进行了扩展介绍，帮助读者了解更规范、更复杂的 JavaScript 程序设计和输入验证判断方法。

4.7 强化练习

学习完本章后，请添加下面功能：

增加手机号码、邮政编号两项输入验证功能，要求使用正则表达式实训，如图 4-4 所示：

图 4-4　扩展练习效果图