第 3 章　JavaScript 基础知识

JavaScript 是目前互联网最流行的脚本语言，于 1995 年由 Netscape 公司的 Brendan Eich 首次设计实现而成。JS 脚本运行在客户端，是一种解释型脚本语言，其功能主要是用来向 XHTML 页面添加交互行为，比如：验证表单中用户名与口令是否正确？检查邮箱地址是否符合标准，实现出生年月日之间的三级联动、检查浏览器类型等，使网页效果更加完美。本章将从以上几个方面来认识 JavaScript 基本语法。

本章知识点：

- JavaScript 语法
- 熟悉 JavaScript 应用

3.1　JavaScript 语法

JavaScript 是一种广泛用于客户端的脚本语言，通过浏览器中的 JavaScript 引擎来解释，目前的统一标准是 ECMAScript。XHTML 实现了网页内容的安排；CSS 与 XHTML 相结合解决了网页整体布局、各部分内容的美化；而 JavaScript 嵌入 XHTML 则增加了静态页面中的动态效果，三者紧密结合使得网页更加生动，因此学习网页设计是离不开 JavaScript 基础语法的。

学习 JavaScript 需要从以下三部分内容着手。

（1）ECMAScript，描述该语言的语法和基本对象。

（2）文档对象模型（Document Object Model，DOM），描述处理网页内容的方法和接口。

（3）浏览器对象模型（Browser Object Model，BOM），描述与浏览器进行交互的方法和接口。

接下来针对以上三部分分别进行详细说明。

3.1.1　ECMAScript

JavaScript 同其他编程语言一样，有它自身的一套语法规则，用来详细说明如何使用该语言进行程序的编写。不同的语言规则稍有不同，要想熟练运用 JavaScript，还需要深入学习 JavaScript 程序设计知识。MAScript 就是指 JavaScript 基本语法部分。

1. 引入JavaScript

JavaScript 与 CSS 在嵌入 XHTML 时稍有不同，引入 CSS 时除内联方式外，外部样式

表和内部样式表都只能在<head>标签内部出现，而 JavaScript 可以嵌入到 XHTML 文档的任意标签位置。只需将 JavaScript 代码使用<script>…</script>声明即可。

1）引入外部文件（推荐）

引入外部文件方式有点儿类似 CSS 外部样式表方式，这种方法首先将 JavaScript 代码集中到一个独立的 js 文件中，然后通过 XHTML 文档使用<script>标签声明引入外部 js 文件来实现 JS 动态效果。此方法与引入 CSS 不同之处在于出现位置可以不固定，可以在<body>内部任意位置，也可以在<head>内部，甚至可以写到 XHTML 整个文档之前，但框架（<frameset>）页面例外。

语法格式如下。

```
<html>
    <head>
        <title>引入外部 js 文件</title>
        <script  src="myScript.js"></script>
    </head>
    <body>
    </body>
</html>
```

将 JavaScript 代码段事先写入指定文件 myScript.js 中，需要使用该文件时只需在 XHTML 文档中引入即可，位置任意。

2）使用内部 js 代码

将 JavaScript 代码集中直接写入到 XHTML 文档内部，位置任意。

语法如下。

```
<html>
    <head>
        <title>引入内部 js 代码</title>
    <script>
    <!--
        (JavaScript 代码段)
    //-->
    </script>
    <body>
    </body>
</html>
```

说明：其中“<!--”与“-->”是 XHTML 中的注释，主要是针对不能识别 JavaScript 代码的浏览器准备的。当浏览器无法解释 JavaScript 代码时，注释符号会将 JavaScript 代码当作注释信息，而不会原原本本显示到客户端的屏幕上，一般也可忽略。

3）写入浏览器地址栏用于调试

JavaScript 代码不仅可以出现在 XHTML 文档中，也可以直接输入到浏览器地址栏里，常用来进行临时调试程序。

语法如下。

```
JavaScript:<JavaScript 语句>
```

例如，打开任意浏览器，找到地址栏位置，输入 JavaScript 调试代码如下。

```
JavaScript:alert("ok");
```

2. 标识符

标识符指的是 JavaScript 中定义的各种符号，例如变量名、函数名等。JavaScript 中标识符的命名规则与其他语言的规则相同，可以包括大小写字母、数字、下划线和美元符号（$）,数字不能作为开头。不能与保留字冲突，且严格区分大小写。

这里重点介绍变量，变量就是用于存储用途的各种类型的数据。

语法如下。

```
var  变量名 [= 变量值];
```

下面通过一个小实例来认识一下 JavaScript 如何使用变量。

3-1.html

```
<html>
 <head>
   <title>变量练习</title>
<script>
//声明变量但不赋值
var  price;
price=30.1;
//声明变量同时赋值
var  name = "zhangsan";
var  age = 20;
var  x=1,y=8;
</script>
</head>
<body>
</body>
</html>
```

3. 数据类型

JavaScritp 脚本语言同其他语言一样，有它自身的基本数据类型，包括：Boolean、Undefined、Number、String、Null、数组和对象。但 JavaScript 也有其特殊的地方，即 JavaScript 的数据类型是动态数据类型，变量的数据类型可根据上下文而变化。

下面通过一个小实例来认识一下 JavaScript 的数据类型。

3-2.html

```
<html>
 <head>
   <title>数据类型练习</title>
<script>

/*
Boolean、Undefined、Number、String、Null、数组和对象。
*/
//只声明但不赋值时变量为未定义类型
//没有生命的变量也是未定义
var  name;                                          //undefined
//数值类型，最基本的数据类型，主要用于完成数学运算
var  age=18;                                        //number
```

```
//布尔类型，只有两个取值：true 和 false，用于表示状态
var  bool = true;                                    //Boolean
//字符串类型用来表示文本的数据类型
var  myName="zhangsan";                              //string
//数组类型
var cars=new Array("Audi","BMW","Volvo");            //数组
cars = null;                                         //清空对象
//null 表示“无值”
//对象
var o = new Object();                                //对象
</script>
</head>
<body>
</body>
</html>
```

4. 运算符

JavaScript 中的运算符与其他语言比较没有任何区别。运算符是一系列用来完成操作的符号，数据通过运算符的连接可以组成各种表达式。根据运算类别，JavaScript 中的运算符主要包括算术运算符、逻辑运算符、字符串运算符以及条件运算符等，具体见表 3-1。

表 3-1　运算符

	[a]	数组的下标
	a=b	把 b 的值赋给 a
	-a	返回 a 的相反数
算术运算符	a*b	返回 a 与 b 的乘积
	a/b	返回整除的商
	a%b	返回 a 除以 b 的余数
	a+b	返回 a 加 b 的值（如果 a 或者 b 为字符串，则表示连接）
	a-b	返回 a 减 b 的值
逻辑运算符	a<b a<=b a>=b a>b a==b a!=b	当符合条件时返回 true 值，否则返回 false 值
	a&&b	当 a 和 b 同时为 true 时返回 true，否则返回 false
	a||b	当 a 和 b 任意一个为 true 时返回 true； 当两者同时为 false 时返回 false
条件运算符	c?a:b	当条件 c 为 true 时返回 a 值，否则返回 b 值
赋值运算符	a+=b a-=b a*=b a/=b a%=b	a 与 b 相加/减/乘/除/求余，所得结果赋给 a

5. 注释

注释是不被执行的代码，注释的出现主要是为了提高代码的可读性。JavaScript 中的注释包括两种：单行注释；多行注释。

下面通过实例来了解一下 JavaScript 中的注释。

3-3.html

```
<html>
 <head>
    <title>注释练习</title>
<script>

//表示单行注释，只为程序员提供信息，不会被执行到
/*
表示多行注释。
要养成写注释的好习惯，当其他人阅读或者自己调试过程中能够节省大量宝贵时间。
*/
</script>
</head>
<body>
</body>
</html>
```

6. 语句结束符

JavaScript 脚本中一行可以写一条语句，也可以写多条语句，每条可执行语句都必须使用“;”作为结束标志，如果没有语句只出现单独的“;”叫作空语句。

语法如下。

```
;
```

或者

```
<语句>;
```

7. 流程控制语句

流程控制语句用来控制程序执行的流程，实现程序的各种结构方式。任何开发语言中最重要的语句莫过于流程控制语句了，没有它们编程人员就无法实现正常的数据判断。JavaScript 中包括以下几种。

1）简单语句

简单语句是指程序自始至终都按照语句在文档中出现的先后顺序执行，也叫顺序结构。下面通过一个实例了解一下简单语句。

3-4.html

```
<html>
 <head>
    <title>简单语句</title>
<script>
var  n=0;
```

```
n=n+1;
alert(n);
</script>
</head>
<body>
</body>
</html>
```

2）条件语句 if

程序编写过程中，有时需要根据不同情况选择执行不同代码，此时可以使用条件语句来完成。JavaScript 中条件语句包括以下两种。

（1）if 语句——根据指定条件是否为 true，选择相应执行代码；

（2）switch 语句——根据条件取值选择多个代码块之一来执行。

本节重点介绍 if 语句。

语法：

```
if (<条件 1>){
    <语句块 1>
}
[else {
    <语句块 2>
}]
```

该语句中带有“[]”的内容是可选部分，表示当条件 1 成立则执行语句块 1 中的代码，否则执行语句块 2 中的代码。其中语句块如果为一条语句，可省略“{}”；多条语句组合而成时，可添加“{}”以增强程序的可读性。

下面通过一个小实例来认识一下 if 语句。

3-5.html

```
<html>
 <head>
   <title>if 语句练习</title>
<script>
//if 简单语句
if(true) alert("ok");                    //输出“ok”
//if…else 语句
if(false){
alert("语句块 1")
}
else{
alert("语句块 2");
}                                        //输出“语句块 2”
//嵌套后形成的 if…else if…else 语句
var a=1;
var b=2;
if (a == 1)
    alert("a=1");
else if (b == 1)
    alert(a+b);
else
    alert(a-b);
//输出：a=1
```

```
//if 语句支持各种嵌套使用
if (a == 1){
    if (b == 1)
        alert(a+b);
}else
    alert(a-b);
//输出：空，因为 b==1 不成立，所以不执行任何语句
</script>
</head>
<body>
</body>
</html>
```

if 语句满足邻近匹配原则，即 else 语句总是和最近的 if 语句相匹配。

3）循环语句

循环语句即周而复始地执行同一段代码。这种结构非常重要，也是计算机最善于执行的一种操作。使用循环语句可以帮助开发人员节省大量的时间。

JavaScript 支持不同类型的循环：

（1）for——循环代码块循环一定的次数。

（2）for/in——循环遍历对象的属性。

（3）while——当指定的条件为 true 时循环指定的代码块。

（4）do/while——同样当指定的条件为 true 时循环指定的代码块。

下面通过一个实例来认识一下循环语句。

3-6.html

```
<html>
 <head>
   <title>循环语句练习</title>
<script>

// 第一种 for 循环
/* 语法：
for (语句 1; 语句 2; 语句 3)
  {
    被执行的代码块
}
其中：
语句 1      <变量>=<初始值>，在循环（代码块）开始前执行，为可选语句；
语句 2      定义运行循环（代码块）的条件，为可选语句；
语句 3      <变量累加方法>，在循环（代码块）已被执行之后执行，为可选语句；
*/
for (var i=0; i<5; i++)
  {
    alert(i);
  }
//输出：01234
/*
 第二种 while 循环
 while (<循环条件>) <语句>;
*/
var i=0;
```

```
while (i<5)
  {
  alert(i);
  i++;
}
//输出：01234
/*
 第三种 do…while 循环
 do{<语句>} while（条件）；
do…while 循环是 while 循环的变体，它与 while 循环最大的区别在于，无论条件是否成立，语句都会被执行至少一次。
*/
var  a=0;
do
  {
  alert(a);
  a++;
  }
while (a<5);
//输出：01234
</script>
</head>
<body>
</body>
</html>
```

4）跳转语句 break 和 continue

跳转语句指的是程序执行循环语句时根据条件终止循环的一种语句。JavaScript 跳转语句包括以下两条语句。

（1）Break 语句——作用是终止后面循环语句的执行并立即跳出当前层循环。

（2）Continue 语句——作用是中止本次循环，立刻执行下一次循环。

下面通过一个小实例来认识一下跳转语句的作用。

3-7.html

```
<html>
 <head>
   <title>break 和 continue 练习</title>
<script>
for (i = 1; i < 5; i++) {
if (i == 3) continue;
alert(i);
}
//输出：124
//尝试运行相同代码段，将 continue 替换成 break 后，则输出结果变为：12
</script>
</head>
<body>
</body>
</html>
```

说明：本段代码中使用 for 循环语句，当 i 取值为 3 时则不再执行本次循环的 alert 语句，而跳转到下一次循环，所以 i=3 不显示。如果将 continue 替换为 break，因为本次循环

为单层循环，则后续所有的循环都终止。当循环进行嵌套时 break 只退出当前层循环，但外层循环不受影响。

8. 对象

JavaScript 将 XHTML 文档中的标签、浏览器的各个属性以及 JavaScript 语法中的各种标识符等所有能涉及的内容都划分成大大小小的对象，所有的编程都是以对象为出发点，基于对象。例如，对象可以是一个变量、一段文字、一幅图片、一个表单（Form）以及屏幕的分辨率等。每个对象有它自己的属性、方法和事件。下面选取几个常用对象进行简单介绍，详细信息可查询在线手册 w3school。

对象操作基本语法：

```
<对象名>.<属性或方法名>
```

1）Number 对象

JavaScript 本身有数字类型，内置 number 对象就是针对原始数值的包装对象。创建 Number 对象的语法如下。

```
var n1 = new  Number(value);
```

Number 对象的常用属性如表 3-2 所示。

表 3-2　Number对象属性

属　　性	描　　述
MAX_VALUE	可表示的最大的数
MIN_VALUE	可表示的最小的数
NaN	非数字值

下面通过具体实例来熟悉一下 Number 对象的属性。

3-8.html

```
<html>
 <head>
   <title>Number 对象练习</title>
<script>
//JavaScript 基本数据类型
var  n1 = 12;
alert(typeof n1);                    //输出 number
//Number 对象
var  n2 = new  Number(3);
alert(typeof n2);                    //输出 object
var s1 = n2.toString();
alert(typeof s1);                    //输出 string
//Number 对象自身属性
alert（Number.MAX_VALUE）;
alert（Number.MIN_VALUE ）;
</script>
</head>
<body>
```

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

2）Array 对象

Array 作为 JavaScript 内置对象，有着非常强大的功能。Array 对象存储量巨大，因此数组可以随意扩容；还可以实现存储类型相同或不同的数据。其创建语法如下。

```
new Array();
new Array(size);
new Array(element0, element1, …, elementn);
```

Array 对象常用属性及方法见表 3-3。

表 3-3　Array 对象属性及方法

属　性	描　述
length	设置或返回数组中元素的数目
方　法	**描　述**
concat()	连接两个或更多的数组，并返回结果
join()	把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔
pop()	删除并返回数组的最后一个元素
push()	向数组的末尾添加一个或更多元素，并返回新的长度
shift()	删除并返回数组的第一个元素
sort()	对数组的元素进行排序
splice()	删除元素，并向数组添加新元素

下面通过一个应用实例来熟悉一下 Array 对象的使用。

3-9.html

```
<html>
 <head>
   <title>Array 对象练习</title>
<script>
//Array 数组对象
var arr1 = new  Array();                //创建一个空数组
arr1[1]= "zhangsan";                    //使用下标方式指定元素
arr1[2]=18;
var arr2 = new  Array(3);               //创建一个 3 个元素的数组
var arr3 = new  Array(1,2,3,4,5,6,7);   //创建数组同时指定元素
alert(arr3.length);                     //输出：7
arr2 = arr1.concat(arr3);               //连接两个数组形成新的数组
alert(arr2.length);                     //输出：10
//隐式声明
var arr4 = [5,4,3,2,1];
//新元素添加到数组结尾，并返回数组新长度
alert(arr4.push(6));
//输出结果：6
//添加新元素到开头
alert(arr4.unshift(6));
//输出结果：7
//移除最后一个元素
var n1 = arr4.pop();
alert(n1);
```

```
//输出结果：6
//删除并返回数组的第一个元素
var n2 = arr4.shift();
alert(n2);
//输出结果：6
//数组元素排序
alert(arr4.sort());
//输出结果：1,2,3,4,5
//数组合并
var arr5 = arr4.concat(arr3);
</script>
</head>
<body>
</body>
</html>
```

3）Boolean 对象

Boolean 对象用于将非逻辑值转换为逻辑值，仅包含两个取值，即“true”和“false”。在 JavaScript 中，0、-0、null、""、false、undefined 或 NaN，其逻辑值都为 false，其他情况逻辑值为 true。该对象的创建语法如下。

```
var bool =  new Boolean(value);
```

下面通过一个应用实例来了解一下布尔对象。

3-10.html

```
<html>
 <head>
   <title>Boolean 对象练习</title>
<script>
//Boolean 布尔对象，该对象表示两个值：“true”或“false”
var b1 = new Boolean(1);
alert(b1);
var b2=new Boolean(0)
var b3=new Boolean(1)
var b4=new Boolean("")
var b5=new Boolean(null)
var b6=new Boolean(NaN)
var b7=new Boolean("false")
document.write("0 的逻辑值是 "+ b2 +"<br />")
//输出结果：0 的逻辑值是 false
document.write("1 的逻辑值是 "+ b3 +"<br />")
//输出结果：1 的逻辑值是 true
document.write("空字符串的逻辑值是 "+ b4 + "<br />")
//输出结果：空字符串的逻辑值是 false
document.write("null 的逻辑值是 "+ b5+ "<br />")
//输出结果： null 的逻辑值是 false
document.write("NaN 的逻辑值是 "+ b6 +"<br />")
//输出结果：NaN 的逻辑值是 false
document.write("字符串 'false'的逻辑值是 "+ b7 +"<br />")
//输出结果：字符串 'false'的逻辑值是 true
</script>
</head>
<body>
```

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

4）Date 对象

Date 对象是 JavaScript 提供的针对日期和时间的操作接口，主要用来处理页面中的时间和日期内容，其创建语法如下。

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

该对象常见方法见表 3-4。

表 3-4　Date 对象方法

方　法	描　述
Date()	返回当日的日期和时间
getDate()	从 Date 对象返回一个月中的某一天（1～31）
getDay()	从 Date 对象返回一周中的某一天（0～6）
getMonth()	从 Date 对象返回月份（0～11）
getFullYear()	从 Date 对象以 4 位数字返回年份
getHours()	返回 Date 对象的小时（0～23）
getMinutes()	返回 Date 对象的分钟（0～59）
getSeconds()	返回 Date 对象的秒数（0～59）
setDate()	设置 Date 对象中月的某一天（1～31）
setMonth()	设置 Date 对象中月份（0～11）
setFullYear()	设置 Date 对象中的年份（4 位数字）
setHours()	设置 Date 对象中的小时（0～23）
setMinutes()	设置 Date 对象中的分钟（0～59）
setSeconds()	设置 Date 对象中的秒钟（0～59）

下面通过一个应用实例来具体认识一下该对象。

3-11.html

```
<html>
 <head>
   <title>Date 对象练习</title>
<script>
//Date 日期对象，主要用于处理日期和时间
//创建日期对象
var  myDate = new Date();
//返回当前周几
alert(myDate.getDay());
//返回当前月份
alert(myDate.getMonth()+1);
//返回 4 位形式当前年份
alert(myDate.getFullYear());
//自定义新的时间
myDate.setDate(2);
myDate.setMonth(3);
//显示自定义后的新时间
alert(myDate);
</script>
</head>
<body>
```

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

5）String 对象

String 对象主要用来生成字符串的包装对象实例，允许操作和格式化文本字符串以及确定和定位字符串中的子字符串。来自客户端网页中的各种信息都是以字符串形式进行收集的，所以在 JavaScript 中处理字符串非常关键。该对象的创建语法如下。

```
var   s1 = new  String(value);
```

String 对象常见属性及方法如表 3-5 所示。

表 3-5　String对象属性及方法

属　　性	描　　述
length	字符串的长度
方　　法	**描　　述**
charAt(number)	返回在指定位置的字符
indexOf(str)	检索字符串
match(str)	找到一个或多个正则表达式的匹配
replace(str)	替换与正则表达式匹配的子串
search(str)	检索与正则表达式相匹配的值
sub()	把字符串显示为下标
substr(n,m)	从起始索引号 n 提取字符串中指定数目 m 的字符
substring(n,m)	提取字符串中两个指定的索引号之间的字符

下面通过一个应用实例来具体认识一下字符串对象。

3-12.html

```
<html>
 <head>
   <title>String 对象练习</title>
<script>
//String 字符串对象，主要用来处理文本
var  s1 = new String("hello world");
//输出该字符串的长度
alert(s1.length);
var  s2 = s1.indexOf("e");
//输出字母 e 在字符串中的索引号
alert(s2);
//将字符串转换成大写后显示
alert(s1.toUpperCase());
var  s3 = s1.substr(2,5);
//从索引号 2 开始提取，提取 5 个字母
alert(s3);
var  s4 = s1.substring(2,5);
//提取索引号 2 到 5 之间的字母
alert(s4);
</script>
</head>
<body>
</body>
</html>
```

6）Math 对象

Math 对象提供了基本数学函数和常数，用于执行各种数学运算任务。该对象与其他几个对象有所不同，Math 对象并不是对象的类，无须创建实例对象，直接将 Math 作为对象就可以调用所有属性和方法。其常用对象方法如表 3-6 所示。

表 3-6　Math对象方法

方　　法	描　　述
abs(x)	返回数的绝对值
ceil(x)	对数进行上舍入
floor(x)	对数进行下舍入
max(x,y)	返回 x 和 y 中的最大值
min(x,y)	返回 x 和 y 中的最小值
pow(x,y)	返回 x 的 y 次幂
random()	返回 0～1 之间的随机数

下面通过一个应用实例来熟悉一下 Math 对象。

3-13.html

```
<html>
 <head>
   <title>Math 对象练习</title>
<script>
/*
Math 数学对象，用于执行数学运算
Math 对象并不是对象的类，因此没有构造函数 Math()，无须创建,直接通过把 Math 作为对象使用就可以调用其所有属性和方法。
*/
//求绝对值
var x = Math.abs(-3);
alert(x);
//求两个数中的最大数
var  a = Math.max(3,5);
alert(a);
//进行下舍入
var  b = Math.floor(3.6);
alert(b);
</script>
</head>
<body>
</body>
</html>
```

7）RegExp 对象

该对象为正则表达式对象，主要进行各种精确条件的匹配，应用简单，但功能非常强大。例如，当检索某个文本时，可以使用一种模式来描述要检索的内容。其中，模式可以是一个单独的字符，或者也可以包含更多的符号。正则表达式还可用于解析、格式检查、替换等。使用时允许规定字符串中的检索位置，以及要检索的字符类型等。

创建该对象的语法如下。

（1）直接量语法：/pattern/attributes

（2）创建 RegExp 对象的语法：new RegExp(pattern, attributes);

常见属性及方法如表 3-7 所示。

表 3-7　RegExp对象属性及方法

属　　性	描　　述
global	RegExp 对象是否具有全局标志 g
ignoreCase	RegExp 对象是否具有忽略大小写标志 i
方　　法	**描　　述**
exec	检索字符串中指定的值。返回找到的值，并确定其位置
test	检索字符串中指定的值。返回 true 或 false

下面通过应用实例来熟悉一下正则表达式的使用。

3-14.html

```
<html>
 <head>
   <title>RegExp 对象练习</title>
<script>
/*
RegExp 正则表达式对象，它是对字符串执行模式匹配的强大工具，有两种语法形式。
（1）直接量语法：
/pattern/attributes
（2）创建 RegExp 对象的语法：
new RegExp(pattern, attributes);
*/
var patt1=/e/;
if(patt1.test("The best things in life are free"))
alert("ok");
else
alert("no");
var patt2=new RegExp("root");
if(patt2.test("The best things in life are free"))
alert("ok");
else
alert("no");
</script>
</head>
<body>
</body>
</html>
```

9. 函数

函数就是一批重复执行的代码块。当进行一项比较复杂的程序设计时，为了保证程序的清晰、易读、易维护等，通常会将多次重复调用的代码块作为一个独立的任务单元来对待，封装成一个函数。JavaScript 中的函数可以避免在页面载入时被执行到，它只能被事件激活或者声明调用函数。在 JavaScript 中提供了大量内置函数，同时也支持用户自定义函数。简单列举几个常用全局函数如表 3-8 所示。

表 3-8　常用全局函数

函　　数	描　　述
eval()	将括号内的字符串当作表达式来执行
isNaN()	如果括号内的值是“Not a Number”则返回 true，否则返回 false
parseInt()	将括号内的内容转换成整数
parseFloat()	将括号内的内容转换成浮点数
toString()	将括号内的内容转换成字符串
escape()	将括号内的内容进行编码
unescape()	是 escape() 的反过程

下面通过应用实例来体验一下函数的使用。

3-15.html

```
<html>
 <head>
   <title>函数练习</title>
<script>
/*
（1）内置函数：即全局对象的方法，全局对象在使用过程中被忽略。
（2）自定义函数：
function  函数名([参数]) {
函数体
[return[ <值>];]
}
*/
//内置函数
if(isNaN("ZHANGSAN")) alert("非数字");
var n1 = 12;
alert(typeof n1);
var n2 = "a12";
alert(typeof n2);
var n3 = "12";
alert(typeof n3);
alert(typeof  parseInt("a12"));
alert(typeof  parseInt("12"));
//自定义函数的使用
function  isEmail(str){
if(/abc@163.com/.test(str))
return true;
else
alert("no");
}
</script>
</head>
<body>
<input type=text onblur=isEmail(this.value)>
</body>
</html>
```

3.1.2　文档对象模型

HTML DOM 定义了所有 HTML 元素对象和属性，以及访问它们的方法。HTML DOM 就是：

（1）HTML 的标准对象模型；

（2）HTML 的标准编程接口；

（3）W3C 标准。

HTML 文档中的所有内容都是节点：

（1）整个文档是一个文档节点；

（2）每个 HTML 元素是元素节点；

（3）HTML 元素内的文本是文本节点；

（4）每个 HTML 属性是属性节点；

（5）注释是注释节点。

通过 HTML DOM 每个节点都可以被获取、修改、添加或删除。常见的 DOM 对象包括：Document、Body、Button、Form、Frame、Frameset、Image、<Input> Button、<Input> Checkbox、<Input> File、<Input> Password、<Input> Radio、<Input> Reset、<Input> Submit、<Input> Text、Link、Meta、Object、Option、Select、Style、Table、Textarea 等。常见 DOM 方法如表 3-9 所示。

表 3-9　DOM常用方法

方　　法	描　　述
getElementById("ID_name")	返回带有指定 ID_name 的元素节点
getElementsByTagName("x")	返回带有指定标签名称的所有元素节点
appendChild()	把新的子节点添加到指定节点
removeChild()	删除子节点
replaceChild()	替换子节点
insertBefore()	在指定的子节点前面插入新的子节点
createAttribute()	创建属性节点
createElement()	创建元素节点
createTextNode()	创建文本节点
getAttribute()	获取节点属性
setAttribute()	设置节点属性

下面简单介绍几个常用对象模型的使用方法。

1．Document 文档对象

每个载入浏览器的 HTML 文档都是一个 Document 对象，通过 Document 对象可以访问到 HTML 页面中的所有元素。

下面演示具体实例。

3-16.html

```
<html>
```

```
 <head>
    <title>document 对象练习</title>
</head>
<body>
<p id="para">通过文档对象可以使用多种方法获取文档中任何元素。但要注意文档的加载顺序，
所以本次的 js 代码放到了最后。</p>
<h1 id="text"></h1>
<script>
document.write("hello Document");
//第一种通过标签名获取，返回为数组
var p1 = document.getElementsByTagName("p");
alert(p1);
//输出节点列表提示
//第二种通过 id 名获取，返回为节点对象
var p2 = document.getElementById("para");
alert(p2);
//输出段落节点对象提示
//创建节点对象
var  text = document.createTextNode("一级标题");
document.getElementById（"text"）.appendChild(text);
//输出"一级标题"
</script>
</body>
</html>
```

2．Form对象

Form 对象就是 HTML 文档中的表单对象，文档中每出现一次<form>就创建一个 Form 对象。Form 对象的常用方法属性见表 3-10。

表 3-10　Form 对象集合

集　　合	描　　述
elements[]	包含表单中所有元素的数组
属　　性	**描　　述**
action	设置或返回表单的 action 属性
id	设置或返回表单的 id
length	返回表单中的元素数目
method	设置或返回将数据发送到服务器的 HTTP 方法
name	设置或返回表单的名称
方　　法	**描　　述**
reset()	把表单的所有输入元素重置为它们的默认值
submit()	提交表单

下面演示具体应用实例。

3-17.html

```
<html>
 <head>
    <title>form 对象练习</title>
</head>
<body>
<form id="myForm">
```

```
 用户名： <input type="text" id="username">
口　令：<input type="password" id="pass">
</form>
<script>
var myForm = document.getElementById("myForm");
for (var i=0;i<myForm.length;i++)
  {
  document.write(myForm.elements[i].type);
  }
alert(myForm.id);
</script>
</body>
</html>
```

3.1.3　浏览器对象模型

与 HTML DOM 对应，BOM 定义了浏览器中各部分对象的属性及其常用方法。BOM 中常见对象主要包括：Window、Navigator、Screen、History、Location。其中，Window 对象代表一个打开的浏览器窗口或一个框架，是全局对象，无须特别声明，直接使用。下面分别简单介绍 BOM 中的对象。

1．Navigator对象

Navigator 对象是 Window 对象的一部分，完整写法为 window.navigator，简写为 navigator。该对象主要包括浏览器的相关信息，常用属性如表 3-11 所示。

表 3-11　Navigator对象

属　　性	描　　述
appCodeName	返回浏览器的代码名
appMinorVersion	返回浏览器的次级版本
appName	返回浏览器的名称
appVersion	返回浏览器的平台和版本信息
browserLanguage	返回当前浏览器的语言
cookieEnabled	返回指明浏览器中是否启用 cookie 的布尔值
cpuClass	返回浏览器系统的 CPU 等级
platform	返回运行浏览器的操作系统平台

下面演示具体应用实例。

3-18.html

```
<html>
 <head>
   <title>Navigator 对象练习</title>
<script>
document.write(navigator.appCodeName);
document.write("<br/>");
  document.write(navigator. appMinorVersion);
document.write("<br/>");
  document.write(navigator. appName);
document.write("<br/>");
```

```
  document.write(navigator. appVersion);
document.write("<br/>");
  document.write(navigator. browserLanguage);
</script>
</head>
<body>
</body>
</html>
```

2．Screen对象

Screen 对象是 Window 对象的一部分，完整写法为 window.screen，简化后为 screen。该对象主要包括客户端浏览器屏幕的相关信息，用来实现优化输出，常用属性如表 3-12 所示。

表 3-12　Screen 对象属性

属　　性	描　　述
availHeight	返回显示屏幕的高度（除 Windows 任务栏之外）
availWidth	返回显示屏幕的宽度（除 Windows 任务栏之外）
height	返回显示器屏幕的高度
width	返回显示器屏幕的宽度

下面演示具体应用实例。

3-19.html

```
<html>
 <head>
   <title>Screen 对象练习</title>
<script>
document.write(screen.availHeight);
document.write("<br/>");
document.write(screen.availWidth);
document.write("<br/>");
document.write(screen.height);
document.write("<br/>");
document.write(screen.width);
</script>
</head>
<body>
</body>
</html>
```

3．Location对象

Location 对象是 Window 对象的一部分，完整写法为 window.location，简化后写为 location。该对象用于获得当前页面的地址（URL）相关信息。常见属性及方法见表 3-13。

表 3-13　location对象属性及方法

属　　性	描　　述
host	设置或返回主机名和当前 URL 的端口号
hostname	设置或返回当前 URL 的主机名
href	设置或返回完整的 URL

续表

属　　性	描　　述
pathname	设置或返回当前 URL 的路径部分
port	设置或返回当前 URL 的端口号
protocol	设置或返回当前 URL 的协议
search	设置或返回从问号（？）开始的 URL（查询部分）
方　　法	**描　　述**
assign()	加载新的文档
reload()	重新加载当前文档
replace()	用新的文档替换当前文档

下面演示具体应用实例。

3-20.html

```
<html>
 <head>
    <title>location 对象练习</title>
<script>
document.write(location.host);
document.write("<br/>");
  document.write(location.hostname);
document.write("<br/>");
  document.write(location.pathname);
document.write("<br/>");
  document.write(location.port);
document.write("<br/>");
  document.write(location.protocol);

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

3.2　JavaScript 与 jQuery

jQuery 是一个兼容多种浏览器的 JavaScript 库，其核心理念是 write less,do more（用最少的代码实现更丰富的效果）。jQuery 的语法设计可以使开发者在实现相同动态效果时比应用 JavaScript 更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、使用 Ajax 以及其他功能方面，代码更简洁、易读。除此以外，jQuery 还提供了 API 让开发者自由编写插件。其模块化的使用方式使开发者可以很轻松就能开发出功能强大的静态或动态网页。jQuery 的优点如下。

（1）兼容多款浏览器；

（2）以 CSS 选择器为基础查找 DOM 对象；

（3）允许自定义插件；

（4）在线学习文档齐全。

下面简单介绍一下 jQuery 的使用步骤，希望对想入门的读者有所帮助。

1. 下载jQuery库

jQuery 库在 jQuery 官网（http://jquery.com/）中单击 Download jQuery，可以下载到不同使用版本。

（1）Production version：用于实际的网站中，已压缩。

（2）Development version：用于测试和开发，未压缩。

2. 引入jQuery库

在 XHTML 文档中使用 jQuery 库来实现动态效果时，只需将 JQuery 库当作外部 js 文件引入 XHTML 文档。

例如：

```
<html>
<head>
<title>如何引入 jQuery 库</title>
<script src="存放 jquery 库的目录名称/jquery.js"></script>
</head>
<body>
</body>
</html>
```

3. jQuery基础语法

jQuery 是针对 HTML 元素编写的，因此可以对 HTML 元素执行各种操作，见表 3-14 所示。

基础语法：

```
$(selector).action()
```

其中：

（1）美元符号（$）定义 jQuery 对象；

（2）选择器（selector）获取 HTML 元素；

（3）jQuery 的 action()执行对元素的操作。

表 3-14 jQuery基本选择器

语　法	描　述
$(this)	当前 HTML 元素
$("p")	所有 <p> 元素
$(".intro")	所有 class="intro" 的元素
$("#intro")	id="intro" 的元素
$("ul li:first")	每个 的第一个 元素
$("[href]")	选取所有带有 href 属性的元素
$("[href='#']")	选取所有带有 href 值等于“#”的元素
$("[href!='#']")	选取所有带有 href 值不等于“#”的元素
$("[href$='.jpg']")	选取所有 href 值以“.jpg”结尾的元素
$("p").css("background-color","red");	CSS 选择器

下面演示具体应用实例。

3-21.html

```
<html>
<head>
<title>比较 jQuery 与 JavaScript</title>
<script src="jquery.js"></script>
</head>
<body>
<p>体验使用 jQuery 获取元素节点与 JavaScript 有什么不同，主要看简化方面。</p>
<h2 class="intro">看 class 属性如何使用 jQuery</h2>
<h1 id="intro">看 id 属性如何使用 jQuery</h2>
<script>
//JavaScript 方式
//获取 p 元素节点,返回节点列表
document.getElementsByTagName("p");
//获取 id=intro 元素节点，返回该节点对象
document.getElementById("intro");
//jQuery 方式
//获取 p 节点元素的文本节点
$("p").innerHTML;
//获取 class=intro 元素节点的文本节点
$(".intro").innerHTML;
//获取 id=intro 元素节点的文本节点
$("#intro").innerHTML;
</script>
</body>
</html>
```

3.3　JavaScript 应用举例

在熟悉了 JavaScript 基本语法之后，本节使用 JavaScript 来实现一个小案例。JavaScript 是在原有的 HTML 基础上增加一些交互行为，使页面更加生动，但起不到真正的安全作用，如果需要解决页面安全方面的问题，还需要继续学习后续章节 PHP 的运用。

1. 案例要求

（1）声明完整的 HTML DTD，体验 DTD 的作用。
（2）实现 HTML 基本文档结构。
（3）网页声明字符编码方式。
（4）注意文档保存编码方式。
（5）实现简单的 form 表单。
（6）针对表单进行正确性验证。
① 姓名不能超过 30 个字符；
② 电话号码必须是数字；
③ 电子邮箱必须符合基本格式要求。

简单表单效果如图 3-1 所示：

图 3-1　简单表单效果

2. 案例分析

（1）XHTML 中的 DTD 规定了网页中的语法，包括严格类型、过渡类型和针对框架集类型三种，本次案例考虑到使用 HTML 的特性以及浏览器的兼容性问题，因此选择过渡类型。

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

（2）XHTML 基本文档结构包括 DTD、HEAD、BODY 三部分。

（3）本案例为防止页面出现乱码，在<head>标记中的<meta>里声明编码方式为 utf-8。

（4）文档存储时选择 utf-8。

（5）Form 表单需要出现两个文本框，一个密码框，一个“提交”按钮。

（6）表单要求：

① 客户输入姓名时只允许为字符，且长度不能超过 30 个，使用正则表达式；

② 电话号码为数字，使用正则表达式；

③ 邮箱需要以一定格式显示，使用正则表达式。

3. 案例实现

代码如下。

3-22.html

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>使用 JavaScript 验证表单的正确性 </title>
<script type="text/JavaScript">
function test(){
//判断姓名长度不能超过 30 个字符
if(document.myForm.username.value.length>30){
alert("不能超过 30 个字符！");
document.myForm.username.focus();
return  false;
}
//判断电话号码是否只包含数字
if((/^[0-9]+$/).test(document.myForm.phone.value)){
alert("ok");}else{
return  false;}
//判断邮箱格式是否正确
if ((/^\w+\@[A-Za-z0-9]+\.[A-Za-z0-9]+$/).search(document.myForm.email.
value) != -1)
alert("ok");
else
return false;
```

```
}
}
</Script>
</head>
<body>
<form name="myForm" onsubmit="return test();">
姓    名: <input type="text" name="username"><br/>
电话号码: <input type="text" name="phone"><br/>
电子邮箱: <input type="text" name="email"><br/>
            
;&nb
sp;   <input type="submit">
</form>
</body>
</html>
```

习题

一、选择题

1. 开发客户端页面动画效果的语言是（　　）。

   A．CSS　　B．JavaScript　　C．XHTML　　D．HTML

2. 节点（元素，标签）属于 JavaScript 中的哪个组成部分（　　）。

   A．ECMAScript　B．DOM　　C．BOM　　D．JScript

3. 载入外部 js 文件描述正确的有（　　）。

   A．不可以载入本站以外的*.js 文件

   B．载入文件方式的好处是可以独立管理一份代码

   C．在 HTML 中，通过<script>标签的 href 属性载入外部*.js 文件

   D．在 HTML 中，通过<script>标签的 src 属性载入外部*.js 文件

4. 在页面里加载当前路径中的 jQuery.js 写法正确的是（　　）。

   A．<script type="text/JavaScript" src="jQuery.js"></script>

   B．<script src="jQuery.js"></script>

   C．<link　href="jQuery.js" />

   D．<script src="jQuery.js">

5. 以下 JavaScript 变量命名格式正确的是（　　）。

   A．1234　　B．-1234　　C．*1234　　D．A_12

6. 以“.js”为文件扩展名的文件是（　　）。

   A．HTML 文件　B．Java 文件　　C．CSS 文件　　D．JavaScript 文件

二、上机操作

1. 使用 JavaScript 数组实现一段文字循环播放。
2. 使用 JavaScript 实现简单滑动导航栏效果。

3. 编写代码实现以下效果：

```
*
***
*****
*******
*********
***********
*************
```