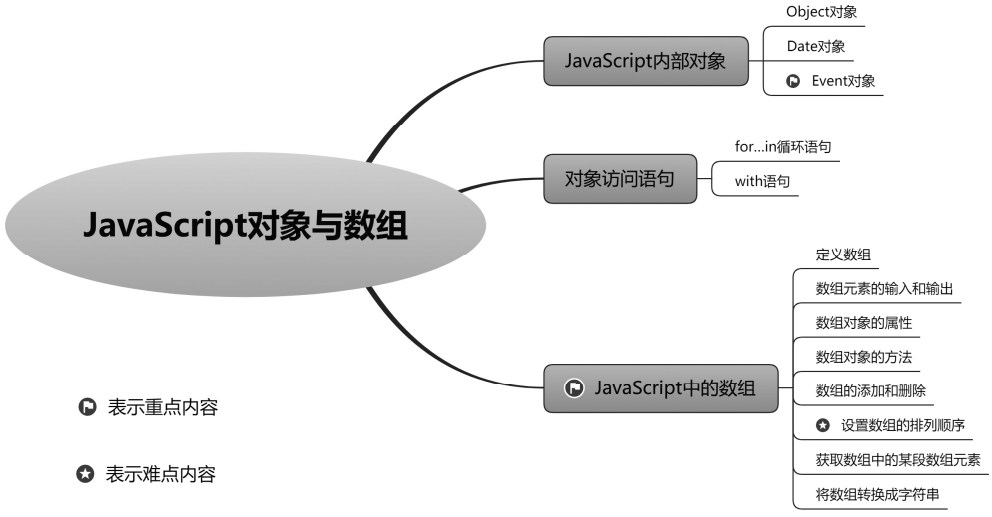
# 第 5 章

# JavaScript 对象与数组

对象是 JavaScript 中的基本数据类型之一，是一种复合的数据类型。它将多种数据类型集中在一个数据单元中，并允许通过对象来存取这些数据的值。本节将对对象的基本概念和基础知识进行简单介绍，通过本章的学习，读者应了解这些对象的简单应用，能够使用访问语句来访问对象，此外还应掌握对数组的简单应用。

本章知识架构及重难点如下。




## 5.1 JavaScript 内部对象

JavaScript 中的内部对象按照使用方式可分为动态对象和静态对象。当引用动态对象的属性和方法时，首先必须使用 new 关键字创建一个对象实例，然后才能使用“对象实例名.成员”的方式来访问其属性和方法；当引用静态对象的属性和方法时，不需要用 new 关键字创建对象实例，直接使用“对象名.成员”的方式来访问其属性和方法即可。

### 5.1.1 Object 对象

Object 对象提供了对象的最基本功能，这些功能构成了其他对象的基础；同时还提供了创建自定

义对象的简单方式，不需要再定义构造函数。使用 Object 对象，可以在程序运行时为 JavaScript 对象随意添加属性，因此可以很容易地创建自定义对象。

### 1．创建 Object 对象

创建 Object 对象的语法格式如下：

```
obj = new Object([value])
```

☑ obj：必选项，表示要赋值为 Object 对象的变量名。

☑ value：可选项，表示任意一种基本数据类型（Number、Boolean 或 String）。如果 value 为一个对象，则返回不做改动的该对象；如果 value 为 null、undefined 或者没有给出，则产生没有内容的对象。

### 2．Object 对象的属性

1）prototype 属性

prototype 属性用于返回对象类型原型的引用。prototype 属性的语法格式如下：

```
objectName.prototype
```

☑ objectName：对象的名称。

用 prototype 属性可以提供对象的类的一组基本功能。对象的新实例将“继承”赋予该对象原型的操作。例如，为数组对象添加返回数组中最大元素值的方法是，声明该函数，然后将它加入 Array.prototype 中，并使用它。代码如下。

```
function array_max(){
    var i, max = this[0];
    for(i = 1; i < this.length; i++)
    {
        if(max < this[i])
        max = this[i];
    }
    return max;
}
Array.prototype.max = array_max;
var x = new Array(1, 2, 3, 4, 5, 6);
var y = x.max();
```

执行上述代码后，y 将保存数组 x 中的最大值，即 6。

JavaScript 固有对象都有只读的 prototype 属性，可以像该例中那样为原型添加功能，但不能被赋予不同的原型。用户定义的对象可以被赋予新的原型。

2）constructor 属性

constructor 属性用于创建对象的函数。constructor 属性的语法格式如下：

```
object.constructor
```

☑ object：必选项，是对象或函数的名称。

constructor 属性是所有具有 prototype 的对象的成员，包括除 Global 和 Math 对象以外的所有 JavaScript 固有对象。constructor 属性保存了对构造特定对象实例的函数的引用，例如：

```
x = new String("Hi");
if(x.constructor == String)
    //进行处理（条件为真）
```

或

```
function MyFunc {
    //函数体
}
y = new MyFunc;
if(y.constructor == MyFunc)
    //进行处理（条件为真）
```

### 3．Object 对象的方法

1）toLocaleString()方法

toLocaleString()方法用于返回一个日期，该日期使用当前区域设置并已被转换为字符串。toLocaleString()方法的语法格式如下：

```
dateObj.toLocaleString()
```

- ☑ dateObj：必选项，可为任意的 Date 对象。
- ☑ toLocaleString()方法返回一个 String 对象，该对象中包含了以当前区域设置的默认格式表示的日期。
- ☑ 公元 1601—1999 年的时间、日期格式，应按照用户控制面板中的区域设置来确定。
- ☑ 此区间外的其他时间，使用 toString()方法的默认格式。

例如，同样是 1 月 5 日，在美国，toLocaleString()可能会返回"01/05/96 00:00:00"；而在欧洲，返回值则可能是"05/01/96 00:00:00"，因为欧洲的惯例是将日期放在月份前面。

**注意**

toLocaleString()通常用来显示结果给用户。不要在脚本中用来做基本计算，因为返回的结果是因机器而异的。

2）toString()方法

toString()方法用于返回对象的字符串表示。toString()方法的语法格式如下：

```
objectname.toString([radix])
```

- ☑ objectname：必选项，表示要得到字符串表示的对象。
- ☑ radix：可选项，指定将数字值转换为字符串时的进制。

toString()方法是所有 JavaScript 内部对象的成员方法，其操作依赖于对象的类型，如表 5.1 所示。

表 5.1　toString()方法的对象及其操作

| 对　象 | 操　作 |
| --- | --- |
| Array | 将 Array 的元素转换为字符串。结果字符串由逗号分隔，且连接起来 |
| Boolean | 如果 Boolean 值是 true，则返回 true；否则，返回 false |
| Date | 返回日期的文字表示法 |

续表

| 对　　象 | 操　　作 |
| --- | --- |
| Error | 返回一个包含相关错误消息的字符串 |
| Function | 返回如下格式的字符串，其中 functionname 是被调用 toString()方法函数的名称：function functionname() { [native code] } |
| Number | 返回数字的文字表示 |
| String | 返回 String 对象的值 |
| 默认 | 返回[object objectname]，其中 objectname 是对象类型的名称 |

3）valueOf()方法

valueOf()方法用于返回指定对象的原始值。valueOf()方法的语法格式如下：

```
object.valueOf()
```

☑ object：必选项，表示任意固有的 JavaScript 对象。不同 JavaScript 固有对象的 valueOf()方法定义也不同，如表 5.2 所示。

表 5.2　JavaScript 固有对象的 valueOf()方法定义

| 对　　象 | 返　回　值 |
| --- | --- |
| Array | 数组的元素被转换为字符串，这些字符串由逗号分隔，并连接在一起。其操作与 Array.toString 和 Array.join 方法相同 |
| Boolean | Boolean 值 |
| Date | 存储的时间是从 1970 年 1 月 1 日午夜开始计的毫秒数 |
| Function | 函数本身 |
| Number | 数字值 |
| Object | 对象本身。这是默认情况 |
| String | 字符串值 |

## 5.1.2　Date 对象

在 Web 开发过程中，可以使用 JavaScript 的 Date 对象（日期对象）实现对日期和时间的控制。例如，想在网页中显示计时时钟，就需要重复生成新的 Date 对象，以获取当前计算机的时间。用户可以使用 Date 对象执行各种日期和时间的操作。

### 1．创建 Date 对象

Date 对象用于对一个对象数据类型求值，该对象主要负责处理与日期和时间有关的数据信息。在使用 Date 对象前，首先要创建该对象。Date 对象语法格式如下：

```
dateObj = new Date()
dateObj = new Date(dateVal)
dateObj = new Date(year, month, date[, hours[, minutes[, seconds[,ms]]]])
```

Date 对象语法中的参数及其说明如表 5.3 所示。

表 5.3　Date 对象的参数及其说明

| 参　数 | 说　明 |
| --- | --- |
| dateObj | 必选项，要赋值为 Date 对象的变量名 |
| dateVal | 必选项，如果是数字值，dateVal 表示指定日期与 1970 年 1 月 1 日午夜之间全球标准时间的毫秒数；如果是字符串，常用的格式为“月 日,年 小时:分钟:秒”，其中月份用英文表示，其余用数字表示，时间部分可以省略；另外，还可以使用“年/月/日 小时:分钟:秒”的格式 |
| year | 必选项，完整的年份，如 1976（而不是 76） |
| month | 必选项，表示月份，是 0～12 的整数（1～12 月） |
| date | 必选项，表示日期，是 1～31 的整数 |
| hours | 可选项，如果提供了 minutes，则必须给出；表示小时，是 0～23 的整数（午夜到 11pm） |
| minutes | 可选项，如果提供了 seconds，则必须给出；表示分钟，是 0～59 的整数 |
| seconds | 可选项，如果提供了 ms，则必须给出；表示秒钟，是 0～59 的整数 |
| ms | 可选项，表示毫秒，是 0～999 的整数 |

下面通过几个示例介绍如何创建日期对象。

例如，返回当前的日期和时间，代码如下。

```
var newDate=new Date();
document.write(newDate);
```

运行结果如下：

```
Wed Jan 27 2021 09:25:16 GMT+0800 (中国标准时间)
```

例如，用年、月、日（2020-12-6）来创建日期对象，代码如下。

```
var newDate=new Date(2020,11,6);
document.write(newDate);
```

运行结果如下：

```
Sun Dec 06 2020 00:00:00 GMT+0800 (中国标准时间)
```

例如，用年、月、日、小时、分钟、秒（2020-12-6 19:18:50）来创建日期对象，代码如下。

```
var newDate=new Date(2020,11,6,19,18,50);
document.write(newDate);
```

运行结果如下：

```
Sun Dec 06 2020 19:18:50 GMT+0800 (中国标准时间)
```

例如，以字符串形式创建日期对象（2020-12-6 19:56:26），代码如下。

```
var newDate=new Date("Dec 6,2020 19:56:26");
document.write(newDate);
```

运行结果如下：

```
Sun Dec 06 2020 19:56:26 GMT+0800 (中国标准时间)
```

### 2．Date 对象的属性

Date 对象的属性包括 constructor 和 prototype，其语法格式与 Object 对象中的 constructor 和 prototype 属性相同。这里介绍这两个属性的用法。

1）constructor 属性

例如，判断当前对象是否为日期对象，代码如下。

```
var newDate=new Date();
if(newDate.constructor==Date)
      document.write("日期型对象");
```

运行结果如下：

```
日期型对象
```

2）prototype 属性

例如，用自定义属性来记录当前日期是本周的周几，代码如下。

```
var newDate=new Date();                  //当前日期为 2021-1-27
Date.prototype.mark=null;                //向对象中添加属性
newDate.mark=newDate.getDay();           //向添加的属性中赋值
alert(newDate.mark);
```

运行结果如下：

```
3
```

### 3．Date 对象的方法

Date 对象是 JavaScript 中的一种内部对象。Date 对象没有可以直接被读写的属性，所有对日期和时间的操作都是通过方法来完成的。Date 对象的方法及其说明如表 5.4 所示。

表 5.4　Date 对象的方法及其说明

| 方　法 | 说　明 |
|---|---|
| Date() | 返回系统当前的日期和时间 |
| getDate() | 从 Date 对象中返回一月中的某一天（1～31） |
| getDay() | 从 Date 对象中返回一周中的某一天（0～6） |
| getMonth() | 从 Date 对象中返回月份（0～11） |
| getFullYear() | 从 Date 对象中返回 4 位数年份 |
| getYear() | 从 Date 对象中以两位或 4 位数字返回年份 |
| getHours() | 返回 Date 对象中的小时数（0～23） |
| getMinutes() | 返回 Date 对象中的分钟数（0～59） |
| getSeconds() | 返回 Date 对象中的秒数（0～59） |
| getMilliseconds() | 返回 Date 对象中的毫秒数（0～999） |
| getTime() | 返回 1970 年 1 月 1 日午夜至今的毫秒数 |

续表

| 方　法 | 说　明 |
| --- | --- |
| getTimezoneOffset() | 返回本地时间与格林尼治标准时间（GMT）的分钟差 |
| getUTCDate() | 根据协调世界时从 Date 对象中返回一月中的一天（1～31） |
| getUTCDay() | 根据协调世界时从 Date 对象中返回一周中的一天（0～6） |
| getUTCMonth() | 根据协调世界时从 Date 对象中返回月份（0～11） |
| getUTCFullYear() | 根据协调世界时从 Date 对象中返回 4 位数的年份 |
| getUTCHours() | 根据协调世界时返回 Date 对象中的小时数（0～23） |
| getUTCMinutes() | 根据协调世界时返回 Date 对象中的分钟数（0～59） |
| getUTCSeconds() | 根据协调世界时返回 Date 对象中的秒数（0～59） |
| getUTCMilliseconds() | 根据协调世界时返回 Date 对象中的毫秒数（0～999） |
| parse() | 返回 1970 年 1 月 1 日午夜到指定日期（字符串）的毫秒数 |
| setDate() | 设置 Date 对象中的日期（1～31） |
| setMonth() | 设置 Date 对象中的月份（0～11） |
| setFullYear() | 设置 Date 对象中的年份（4 位数字） |
| setYear() | 设置 Date 对象中的年份（两位或 4 位数字） |
| setHours() | 设置 Date 对象中的小时数（0～23） |
| setMinutes() | 设置 Date 对象中的分钟数（0～59） |
| setSeconds() | 设置 Date 对象中的秒数（0～59） |
| setMilliseconds() | 设置 Date 对象中的毫秒数（0～999） |
| setTime() | 使用毫秒数设置日期和时间 |
| setUTCDate() | 根据协调世界时设置 Date 对象中的日期（1～31） |
| setUTCMonth() | 根据协调世界时设置 Date 对象中的月份（0～11） |
| setUTCFullYear() | 根据协调世界时设置 Date 对象中的年份（4 位数字） |
| setUTCHours() | 根据协调世界时设置 Date 对象中的小时数（0～23） |
| setUTCMinutes() | 根据协调世界时设置 Date 对象中的分钟数（0～59） |
| setUTCSeconds() | 根据协调世界时设置 Date 对象中的秒数（0～59） |
| setUTCMilliseconds() | 根据协调世界时设置 Date 对象中的毫秒数（0～999） |
| toSource() | 代表对象的源代码 |
| toString() | 把 Date 对象转换为字符串 |
| toTimeString() | 把 Date 对象中的时间部分转换为字符串 |
| toDateString() | 把 Date 对象中的日期部分转换为字符串 |
| toGMTString() | 根据格林尼治时间，把 Date 对象转换为字符串 |
| toUTCString() | 根据协调世界时，把 Date 对象转换为字符串 |
| toLocaleString() | 根据本地时间格式，把 Date 对象转换为字符串 |
| toLocaleTimeString() | 根据本地时间格式，把 Date 对象中的时间部分转换为字符串 |
| toLocaleDateString() | 根据本地时间格式，把 Date 对象中的日期部分转换为字符串 |
| UTC() | 根据协调世界时，获得一个日期，然后返回 1970 年 1 月 1 日午夜到该日期的毫秒数 |
| valueOf() | 返回 Date 对象的原始值 |

**误区警示**

因为使用 getMonth()方法获取的月份比系统中实际月份的值小 1，所以在使用 getMonth()方法获取当前月份的值时要加上 1。代码如下。

```
var date = new Date();                          //创建当前日期对象
alert("现在是："+(date.getMonth()+1)+"月");      //输出现在的月份
```

**【例 5.1】**输出当前的日期和时间。**（实例位置：资源包\TM\sl\5\01）**

应用 Date 对象中的方法获取当前完整的年份、月份、日期、星期、小时数、分钟数和秒数，将当前的日期和时间分别连接在一起并输出。代码如下。

```
var now=new Date();                                    //创建日期对象
var year=now.getFullYear();                            //获取当前完整的年份
var month=now.getMonth()+1;                            //获取当前月份
var date=now.getDate();                                //获取当前日期
var day=now.getDay();                                  //获取当前星期
var week="";                                           //初始化变量
switch(day){
    case 1:                                            //如果变量 day 的值为 1
        week="星期一";                                  //为变量赋值
        break;                                         //退出 switch 语句
    case 2:                                            //如果变量 day 的值为 2
        week="星期二";                                  //为变量赋值
        break;                                         //退出 switch 语句
    case 3:                                            //如果变量 day 的值为 3
        week="星期三";                                  //为变量赋值
        break;                                         //退出 switch 语句
    case 4:                                            //如果变量 day 的值为 4
        week="星期四";                                  //为变量赋值
        break;                                         //退出 switch 语句
    case 5:                                            //如果变量 day 的值为 5
        week="星期五";                                  //为变量赋值
        break;                                         //退出 switch 语句
    case 6:                                            //如果变量 day 的值为 6
        week="星期六";                                  //为变量赋值
        break;                                         //退出 switch 语句
    default:                                           //默认值
        week="星期日";                                  //为变量赋值
        break;                                         //退出 switch 语句
}
var hour=now.getHours();                               //获取当前小时数
var minute=now.getMinutes();                           //获取当前分钟数
var second=now.getSeconds();                           //获取当前秒数
//为字体设置样式
document.write("<span style='font-size:24px;'>");
```

```
document.write("今天是："+year+"年"+month+"月"+date+"日 "+week);    //输出当前的日期和星期
document.write("<br>现在是："+hour+":"+minute+":"+second);          //输出当前的时间
document.write("</span>");                                          //输出</span>结束标记
```

运行结果如图 5.1 所示。


图 5.1　输出当前的日期和时间

应用 Date 对象的方法除了可以获取日期和时间之外，还可以设置日期和时间。在 JavaScript 中只要定义了一个日期对象，就可以针对该日期对象的日期部分或时间部分进行设置。例如，设置指定日期和时间的代码如下。

```
var myDate=new Date();                  //创建当前日期对象
myDate.setFullYear(2021);               //设置完整的年份
myDate.setMonth(5);                     //设置月份
myDate.setDate(12);                     //设置日期
myDate.setHours(10);                    //设置小时数
myDate.setMinutes(10);                  //设置分钟数
myDate.setSeconds(10);                  //设置秒数
document.write(myDate);                 //输出日期对象
```

运行结果如下。

```
Sat Jun 12 2021 10:10:10 GMT+0800 (中国标准时间)
```

在脚本编程中可能需要处理许多对日期的计算，例如计算经过固定天数或星期之后的日期或计算两个日期之间的天数。在这些计算中，JavaScript 日期值都是以毫秒为单位的。

**【例 5.2】获取当前日期距离明年元旦的天数。（实例位置：资源包\TM\sl\5\02）**

应用 Date 对象中的方法获取当前日期距离明年元旦的天数。代码如下。

```
var date1=new Date();                                   //创建当前的日期对象
var theNextYear=date1.getFullYear()+1;                  //获取明年的年份
date1.setFullYear(theNextYear);                         //设置日期对象 date1 中的年份
date1.setMonth(0);                                      //设置日期对象 date1 中的月份
date1.setDate(1);                                       //设置日期对象 date1 中的日期
var date2=new Date();                                   //创建当前的日期对象
var date3=date1.getTime()-date2.getTime();              //获取两个日期相差的毫秒数
var days=Math.ceil(date3/(24*60*60*1000));              //将毫秒数转换成天数
alert("今天距离明年元旦还有"+days+"天");                  //输出结果
```

运行结果如图 5.2 所示。

在 Date 对象的方法中还提供了一些以 to 开头的方法，这些方法可以将 Date 对象转换为不同形式的字符串。示例代码如下。

```
<h3>将 Date 对象转换为不同形式的字符串</h3>
<script type="text/javascript">
var newDate=new Date();                                 //创建当前日期对象
document.write(newDate.toString()+"<br>");              //将 Date 对象转换为字符串
document.write(newDate.toTimeString()+"<br>");          //将 Date 对象的时间部分转换为字符串
document.write(newDate.toDateString()+"<br>");          //将 Date 对象的日期部分转换为字符串
document.write(newDate.toLocaleString()+"<br>");        //将 Date 对象转换为本地格式的字符串
//将 Date 对象的时间部分转换为本地格式的字符串
```

```
document.write(newDate.toLocaleTimeString()+"<br>");
//将 Date 对象的日期部分转换为本地格式的字符串
document.write(newDate.toLocaleDateString());
</script>
```

运行结果如图 5.3 所示。


图 5.2　输出当前日期距离明年元旦的天数


图 5.3　将日期对象转换为不同形式的字符串

## 5.1.3　Event 对象

JavaScript 的 Event 对象用来描述 JavaScript 的事件。Event 对象代表事件状态，如事件发生的元素、键盘状态、鼠标位置和鼠标按钮状态。一旦事件发生，就会生成 Event 对象。例如，单击一个按钮，浏览器的内存中就会产生相应的 Event 对象。

在 W3C 事件模型中，需要将 Event 对象作为一个参数传递到事件处理函数中。Event 对象也可自动作为参数传递，这取决于事件处理函数与对象绑定的方式。

如果使用原始方法将事件处理函数与对象绑定（通过元素标记的一个属性），则必须把 Event 对象作为参数进行传递，例如：

```
onKeyUp="example(event)"
```

这是 W3C 模型中唯一可像全局引用一样明确引用 Event 对象的方式。这个引用只作为事件处理函数的参数，在别的内容中不起作用。如果有多个参数，则 Event 对象的引用可以以任意顺序排列，例如：

```
onKeyUp="example(this,event)"
```

与元素绑定的函数定义中，应该有一个参数变量来捕获 Event 对象参数，例如：

```
function example(widget,evt){…}
```

此外，还可以通过其他方式将事件处理函数绑定到对象，将这些事件处理函数的引用赋给文档中所需的对象，例如：

```
document.forms[0].someButton.onkeyup=example;
document.getElementById("myButton").addEventListener("keyup",example,false);
```

通过这些方式进行事件绑定，可以防止自己的参数直接到达调用的函数中，但是，W3C 浏览器自动传递 Event 对象的引用并将它作为唯一参数。这个 Event 对象是为响应激活事件的用户或系统行为而创建的，也就是说，函数需要用一个参数变量来接收所传递的 Event 对象。例如：

```
function example(evt){…}
```

事件对象包含作为事件目标的对象（如包含表单控件对象的表单对象）的引用，以便可以访问该对象的任何属性。

### 3．Event 对象的属性

1）altKey 属性

altKey 属性用于返回当事件被触发时，Alt 键是否被按下。返回值为 true 表示被按下，为 false 表示未被按下。altKey 属性的语法格式如下。

```
event.altKey
```

由于 altKey 属性是 Boolean 值，因此可以将该属性应用到 if 语句中，根据获取的值执行不同的操作。

2）ctrlKey 属性

ctrlKey 属性用于返回当事件被触发时，Ctrl 键是否被按下。如果返回值为 true，则表示被按下；如果返回值为 false，则表示未被按下。ctrlKey 属性的语法格式如下。

```
event.ctrlKey
```

由于 ctrlKey 属性是 Boolean 值，因此可以将该属性应用到 if 语句中，根据获取的值执行不同的操作。

3）shiftKey 属性

shiftKey 属性用于返回当事件被触发时，Shift 键是否被按下。如果返回值为 true，则表示被按下；如果返回值为 false，则表示未被按下。shiftKey 属性语法格式如下。

```
event.shiftKey
```

上述 3 个属性的值都是 Boolean 类型的，因此可以将它们组合成一个条件在 if 语句中应用。

**【例 5.3】**判断 Ctrl 键、Shift 键和 Alt 键是否同时被按下。**（实例位置：资源包\TM\sl\5\03）**

下面将 altKey、ctrlKey 和 shiftKey 属性组成一个综合的条件，应用 if 语句判断当 Ctrl 键、Shift 键和 Alt 键同时被按下时执行一个操作。运行结果如图 5.4 所示。


图 5.4　altKey、ctrlKey 和 shiftKey 属性的综合应用

在本例中，创建一个自定义函数 example()，应用 if 语句判断 Ctrl 键、Shift 键和 Alt 键是否同时被按下。如果是，则执行 alert 中的内容并跳转到一个新的文件中。代码如下。

```
<script type="text/javascript">
function example(e){                                    //创建自定义函数
    //应用 if 语句判断 Ctrl 键、Shift 键和 Alt 键是否同时被按下
  if(e.ctrlKey && e.altKey && e.shiftKey){
    //如果 Ctrl 键、Shift 键和 Alt 键同时被按下,则执行下面的内容
      alert('明日科技给您拜年了!');                     //弹出一个对话框
      window.location.href='index_ok.html';             //跳转到一个文件中
  }
}
document.onkeydown=example;                             //应用 onkeydown 事件调用自定义函数 example
</script>
```

4）button 属性

button 属性用于获取事件发生时用户按的鼠标键。该属性的语法格式如下。

```
event.button
```

button 属性的值和说明：0 为按鼠标左键，1 为按鼠标中键，2 为按鼠标右键。

**【例 5.4】右击链接到指定网站。（实例位置：资源包\TM\sl\5\04）**

在浏览网页的过程中，经常会应用鼠标右键进行快捷键操作。为了提高网站的安全性和有效地利用资源，可以在页面中添加右击自动链接到指定网站的功能。在本例中，实现在页面中的任何位置右击都将弹出“明日科技欢迎您”对话框，单击“确定”按钮后将进入明日科技网站的首页中。运行结果如图 5.5 所示。

这里主要应用 Event 对象中的 button 属性，以判断当鼠标右键被按时弹出提示对话框，单击“确定”按钮后跳转到明日科技网站的首页。程序代码如下。

```
<script type="text/javascript">
function gosite(e){
    if(e.button==2){
        alert('明日科技欢迎您');
          window.open("http://www.mingrisoft.com");
          return false;
    }
}
document.onmousedown=gosite;
</script>
```

5）clientX 属性

clientX 属性用于获取鼠标在浏览器窗口中的 *X* 坐标。这是一个只读属性，即只能获取鼠标的当前位置，不能改变鼠标的位置。clientX 属性的语法格式如下。

```
event.clientX
```

6）clientY 属性

clientY 属性用于获取鼠标在浏览器窗口中的 *Y* 坐标。这是一个只读属性，即只能获取鼠标的当前位置，不能改变鼠标的位置。clientY 属性的语法格式如下。

```
event.clientY
```

**【例 5.5】文字跟随鼠标移动。（实例位置：资源包\TM\sl\5\05）**

设计一个文字跟随鼠标移动的例子。当鼠标指针移动到文字上方时，拖曳鼠标即可使工作区中的文字跟随鼠标指针移动。运行结果如图 5.6 所示。


图 5.5　右击弹出提示信息


图 5.6　跟随鼠标移动的文字

本例中，应用 Document 对象中的 onmousedown、onmouseup 和 onmousemove 事件来对文字进行移动控制，应用 Event 对象中的 clientX 和 clientY 属性获取鼠标在当前工作区中的位置，应用 Event 对象中的 target 属性获取触发当前事件的元素，代码如下。

```
<script type="text/javascript">
var z,x,y;                                          //定义变量
function move(e){                                   //创建函数
    z.style.left=temporarily1+e.clientX-x+"px";     //设置文字的位置
    z.style.top=temporarily2+e.clientY-y+"px";      //设置文字的位置
    return false;
}
function down(e){                                   //创建自定义函数，实现文字的移动
    if(e.target.className=="move_out"){             //应用 target 属性获取当前事件的对象
        z=e.target;
        temporarily1=z.offsetLeft;
        temporarily2=z.offsetTop;
        x=e.clientX;                                //获取鼠标在窗口中的 X 位置
        y=e.clientY;                                //获取鼠标在窗口中的 Y 位置
        document.onmousemove=move;                  //当鼠标被移动时执行 move()函数
        document.onmouseup=function(){
            document.onmousemove = null;
        }
     }
}
document.onmousedown=down;                          //当鼠标被按下时执行 down()函数
</script>
<body>
<span class="move_out">吉林省明日科技</span>
</body>
```

7）X 属性

X 属性用于设置或获取鼠标指针位置相对于 CSS 属性中有 position 属性的上级元素的 *X* 轴坐标。如果 CSS 属性中没有 position 属性的上级元素，则默认以 body 元素作为参考对象。X 属性的语法格式如下：

```
event.X
```

如果鼠标事件被触发，鼠标指针被移出窗口外，则返回值为-1。这是一个只读属性，只能通过它获取鼠标的当前位置，但不能用它来更改鼠标的位置。

8）Y 属性

Y 属性用于设置或获取鼠标指针位置相对于 CSS 属性中有 position 属性的上级元素的 *Y* 轴坐标。如果 CSS 属性中没有 position 属性的上级元素，则默认以 body 元素作为参考对象。Y 属性的语法格式如下：

```
event. Y
```

如果鼠标事件被触发，鼠标指针被移出窗口外，则返回值为-1。这是一个只读属性，只能通过它获取鼠标的当前位置，但不能用它来更改鼠标的位置。

**【例 5.6】**创建可移动可关闭的窗口。**（实例位置：资源包\TM\sl\5\06）**

在本例中创建一个可以在页面中移动的窗口，并可对窗口执行关闭操作。运行结果如图 5.7 所示。

本例在用鼠标按住要移动的层时，将层的名称赋给了一个变量，并用该变量对层进行移动的操作。如果直接用层的名称来对层进行移动，则将会选中层后面的一些信息。其中通过 Event 对象中的 X 和 Y 属性在要移动的层上拖曳鼠标来改变当前层的位置。

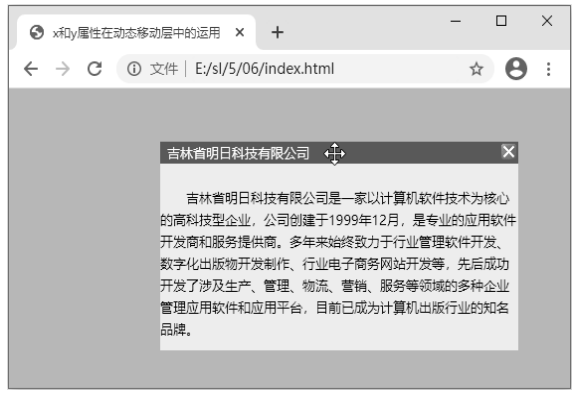



图 5.7　X 和 Y 属性在动态移动层中的运用

（1）在页面中添加两个层，并且其中一个层包含另一个层。在 div1 层的鼠标按下事件 onmousedown 被触发时调用自定义函数 div_down()，代码如下。

```
<div id="div1" style="visibility:visible ; background-color:#FFFF00; position: absolute; top: 60px; left: 200px;
width: 360px; height: 200px;">
    <div id="title" onmousedown="div_down('div1',event)" style="background-color:#30608F;padding:2px;
font-size:13px;text-indent:5px;color:#FFFFFF;cursor:move">吉林省明日科技有限公司</div>
    <img id="close" onclick="hide('div1')" style="position: absolute; right: 2px; top: 2px" src="close.gif" width="15"
height="15">
    <span class="STYLE1"><br>
      吉林省明日科技有限公司是一家以计算机软件技术为核心的高科技型企业，公司创建于 1999
年 12 月，是专业的应用软件开发商和服务提供商。多年来始终致力于行业管理软件开发、数字化出版物开发制作、
行业电子商务网站开发等，先后成功开发了涉及生产、管理、物流、营销、服务等领域的多种企业管理应用软件
和应用平台，目前已成为计算机出版行业的知名品牌。
    </span>
</div>
```

（2）编写用于实现动态移动层的 JavaScript 代码。创建自定义函数 div_down()，获取当前鼠标与层左边界和上边界的距离，代码如下。

```
<script type="text/javascript">
var Obj="none";
var pX;
var pY;
document.onmousemove=div_move;
document.onmouseup=div_up;
function div_down(tag,e){
    Obj=tag;
    pX=e.x-parseInt(document.getElementById(Obj).style.left);
    pY=e.y-parseInt(document.getElementById(Obj).style.top);
}
```

（3）创建自定义函数 div_move()，用于移动层的位置，代码如下。

```
function div_move(e){
    if(Obj!="none"){
        document.getElementById(Obj).style.left=e.x-pX+"px";
        document.getElementById(Obj).style.top=e.y-pY+"px";
```

```
        e.returnValue=false;
    }
}
```

（4）创建自定义函数 div_up()，当松开鼠标时释放对当前层的控制，代码如下。

```
function div_up(){Obj="none";}
```

创建自定义函数 hide()，以动态效果隐藏层，代码如下。

```
function hide(divid){
    document.getElementById(divid).style.visibility = "hidden";
}
```

（5）在页面上移动鼠标时，通过页面的 onmousemove 事件调用自定义函数 div_move()，代码如下。

```
document.onmousemove=div_move;
```

（6）在页面上松开鼠标时，通过页面的 onmouseup 事件调用自定义函数 div_up()，代码如下。

```
document.onmouseup=div_up;
```

9）cancelBubble 属性

cancelBubble 属性检测是否接收上层元素的事件控制。如果该属性的值是 false，则允许被上层元素的事件控制；如果该属性的值为 true，则不被上层元素的事件控制。cancelBubble 属性的语法格式如下：

```
event.cancelBubble[ = cancelBubble]
```

cancelBubble 属性的值是一个可读写的布尔值，默认值为 false。

10）target 属性

target 属性用于获取触发事件的对象，如生成事件的元素、文档或窗口。target 属性的语法格式如下：

```
event.target
```

通过 target 属性可以读、写属于该元素的属性，并调用该元素的任何方法。

**【例 5.7】**单击单元格变色。**（实例位置：资源包\TM\sl\5\07）**

在浏览网页中的表格数据时，有时会忘记浏览到的位置。这时可以用鼠标选中已经读取到的数据，并改变表格中单元格的背景颜色，从而记录数据读取的位置，便于以后继续浏览。运行结果如图 5.8 所示。


图 5.8　选中表格中的单元格

该功能的实现主要通过 onclick 事件，应用 Event 对象中的 target 属性获取发生事件的文档元素，并将其保存到变量 e 中；然后应用 e.tagName（获取当前的标签名称）判断发生事件的文档元素是否在表格的单元格上，并用变量 e 的 style 样式的 color 和 backgroundColor 属性来改变当前单元格的前景色和背景色，其中还应用了 Event 对象中的 cancelBubble 属性，防止向下一个外层对象冒泡；最后应用了 Window 对象中的 lastSelection 属性，获取最后一次选中的单元焦点。

（1）创建自定义函数 select()，保存发生事件的文档元素信息，代码如下。

```
<script type="text/javascript">
var lastSelection = null;
function select(event){
    var e, r, c;
    e = event.target;                    //获取触发事件的元素
    if(e.tagName == "TD"){
        c = findcell(e);
        if(c != null){
            if(lastSelection != null){
                deselectroworcell(window.lastSelection);
            }
            selectroworcell(c);
            lastSelection = c;
        }
    }
    event.cancelBubble = true;      //取消冒泡语句，用于防止向下一个外层对象冒泡
}
table1.onclick=select;
```

（2）创建自定义函数 findcell()，用于判断选中的位置是否是单元格，代码如下。

```
function findcell(e){
    if(e.tagName == "TD"){
        return e;
    }else if(e.tagName == "BODY"){
        return null;
    }else{
        return findcell(e.parentElement);
    }
}
```

（3）创建自定义函数 selectroworcell()，用于改变单元格的前景色和背景色，代码如下。

```
function selectroworcell(r){
    r.style.backgroundColor="darkblue";
    r.style.color="white";
}
```

（4）创建自定义函数 deselectroworcell()，用于将前景色和背景色恢复正常，代码如下。

```
function deselectroworcell(r){
    r.style.backgroundColor = "";
    r.style.color = "";
}
</script>
```

**编程训练（答案位置：资源包\TM\sl\5\编程训练\）**

**【训练 1】**计算两个日期的间隔小时数　将 2021 年 5 月 1 日作为开始日期，将 2021 年 6 月 1 日作为结束日期，计算开始日期和结束日期之间的间隔小时数。

**【训练 2】**高考倒计时　以 2021 年高考时间作为倒计时，在页面中显示距 2021 年高考还有多少天的提示信息。

【训练 3】判断单击的鼠标按键　在页面中单击鼠标弹出一个对话框，在该对话框中显示刚刚单击了哪个鼠标按键。（提示：通过 button 属性进行判断）

# 5.2　对象访问语句

在 JavaScript 中，for…in 循环语句和 with 语句都是专门应用于对象的语句。下面对这两个语句分别进行介绍。

## 5.2.1　for…in 循环语句

for…in 循环语句和 for 循环语句十分相似，用来遍历对象的所有属性。每次都将属性名作为字符串，保存到变量中。for…in 循环语句语法格式如下：

```
for(variable in object ) {
    …statement
}
```

在上述语法中，variable 是一个变量名，声明一个变量的 var 语句、数组的一个元素或者对象的一个属性（它应该是一个适用于赋值表达式左边的值）；object 是一个对象名，或者是计算结果为对象的表达式；statement 通常是一个原始语句或者语句块，由它构成循环的主体。

下面应用 for…in 循环语句输出对象中的属性名和值。首先创建一个对象，并且指定对象的属性，然后应用 for…in 循环语句输出对象的所有属性和值。代码如下：

```
var objectes={user:"×××",age:23,QQ:"1838**",e_mail:"ppp***@sina.com"};
for(var example in objectes){                                    //循环语句
    document.write("属性："+example+"="+objectes[example]+"<br>")
}
```

运行结果如下：

```
属性：user=×××
属性：age=23
属性：QQ=1838**
属性：e_mail=ppp***@sina.com
```

应用 for…in 循环语句可以为对象的每个属性执行一个语句集合，可以对 JavaScript 中的任何对象应用 for…in 循环语句。

## 5.2.2　with 语句

with 语句用于需要多次访问某个对象的属性或方法的情况下，可避免重复使用指定对象的引用。

with 语句语法格式如下：

```
with(object){
    statements
}
```

☑ object：指定 statements 块中没有引用对象的情况下使用哪个对象的引用。

在一个连续的程序代码中，如果多次使用某个对象的多个属性或方法，那么只要在 with 关键字后的括号()中写出该对象实例的名称，就可以在随后的大括号{}中的程序语句中直接引用该对象的属性名或方法名，不必再在每个属性名或方法名前都加上对象实例名和“.”。

例如，应用 with 语句实现 student 对象的多次引用，代码如下。

```
function Student(name,sex,age){
    this.name = name;                                          //设置对象的 name 属性
    this.sex = sex;                                            //设置对象的 sex 属性
    this.age = age;                                            //设置对象的 age 属性
}
var student=new Student("周星星","男",26);                      //创建新对象
with(student){                                                 //应用 with 语句
    alert("姓名："+name+"\n 性别："+sex+"\n 年龄："+age);       //输出多个属性的值
}
```

运行结果如图 5.9 所示。


图 5.9　with 语句的应用

# 5.3　JavaScript 中的数组

可以把数组看作一张单行表格，该表格的每一个单元格又都可以存储一个数据，而且各单元格中存储的数据类型可以不同。这些单元格被称为数组元素，每个数组元素都有一个索引号，通过索引号可以方便地引用数组元素。数组是 JavaScript 中唯一用来存储和操作有序数据集的数据结构。

## 5.3.1　定义数组

在 JavaScript 中数组也是一种对象，这种对象被称为数组对象。因此在定义数组时，也可以使用构造函数。JavaScript 中定义数组的方法主要有 4 种。

### 1．定义空数组

使用不带参数的构造函数可以定义一个空数组。顾名思义，空数组中是没有数组元素的，可以在

定义空数组后再向数组中添加数组元素。其语法格式如下：

```
arrayObject = new Array()
```

☑ arrayObject：必选项。新创建的数组对象名。

例如，创建一个空数组，然后向该数组中添加数组元素，代码如下。

```
var arr = new Array();                    //定义一个空数组
arr[0] = "a";                             //向数组中添加第一个数组元素
arr[1] = "b";                             //向数组中添加第二个数组元素
arr[2] = "c";                             //向数组中添加第三个数组元素
```

在上述代码中定义了一个空数组，此时数组中元素的个数为 0。在为数组的元素赋值后，数组中才有了数组元素。

**误区警示**

定义的数组对象名不能和已存在的变量重名。例如编写如下代码：

```
var user = "明日科技";                    //定义变量 user
var user = new Array();                   //定义一个空数组 user
user[0] = "张三";                         //向数组中添加数组元素
user[1] = "李四";                         //向数组中添加数组元素
document.write(user);                     //输出 user 的值
```

上述代码中，由于定义的数组对象名和已存在的变量重名，变量的值被数组的值所覆盖，因此在输出 user 变量时只能输出数组的值。

### 2．指定数组长度

在定义数组的同时可以指定数组元素的个数。此时并没有为数组元素赋值，所有数组元素的值都是 undefined。其语法格式如下：

```
arrayObject = new Array(size)
```

☑ arrayObject：必选项。新创建的数组对象名。

☑ size：设置数组的长度。由于数组的下标是从零开始的，创建元素的下标将为 0～size−1。

例如，创建一个数组元素个数为 3 的数组，并向该数组中存入数据，代码如下。

```
var arr = new Array(3);                   //定义一个元素个数为 3 的数组
arr[0] = 1;                               //为第一个数组元素赋值
arr[1] = 2;                               //为第二个数组元素赋值
arr[2] = 3;                               //为第三个数组元素赋值
```

在上述代码中定义了一个元素个数为 3 的数组。在为数组元素赋值之前，这 3 个数组元素的值都是 undefined。

### 3．指定数组元素

在定义数组的同时可以直接给出数组元素的值。此时，数组的长度就是在括号中给出的数组元素的个数。其语法格式如下：

```
arrayObject = new Array(element1, element2, element3, ...)
```

☑　arrayObject：必选项。新创建的数组对象名。

☑　element：存入数组中的元素。使用该语法时必须有一个以上元素。

例如，创建数组对象的同时，向该对象中存入数组元素，代码如下。

```
var arr = new Array(123, "零基础学 JavaScript", true);                    //定义一个包含 3 个元素的数组
```

#### 4．直接定义数组

在 JavaScript 中还有一种定义数组的方式，这种方式不需要使用构造函数，直接将数组元素放在一个中括号中，元素与元素之间用逗号分隔。其语法格式如下：

```
arrayObject = [element1, element2, element3, ...]
```

☑　arrayObject：必选项。新创建的数组对象名。

☑　element：存入数组中的元素。使用该语法时必须有一个以上元素。

例如，直接定义一个含有 3 个元素的数组，代码如下。

```
var arr = [123, "零基础学 JavaScript", true];                    //直接定义一个包含 3 个元素的数组
```

### 5.3.2　数组元素的输入和输出

#### 1．数组元素的输入

当向数组对象中输入数组元素时，有 3 种方法，分别介绍如下。

1）在定义数组对象时直接输入数组元素

该方法只能在数组元素确定的情况下才可以使用。例如，在创建数组对象的同时存入字符串数组，代码如下。

```
arrayObj = new Array("a","b","c","d")
```

2）利用数组对象的元素下标向其输入数组元素

该方法可以随意地为数组对象中的各元素赋值，或是修改数组中的任意元素值。例如，在创建一个长度为 7 的数组对象后，向下标为 3 和 4 的元素赋值。

```
arrayObj = new Array(7)
arrayObj[3] = "a";
arrayObj[4] = "b";
```

3）利用 for 语句向数组对象中输入数组元素

该方法主要用于批量向数组对象中输入数组元素，一般用于为数组对象赋初值。例如，可以通过改变变量 n 的值（必须是数值型），给数组对象 arrayObj 赋指定个数的数值元素，代码如下。

```
Var n=7
arrayObj = new Array()
for(var i=0;i<n;i++){
      arrayObj[i]=i
}
```

例如，给指定元素个数的 Array 对象赋值，代码如下。

```
arrayObj = new Array(7)
for(var i=0;i<arrayObj.length;i++){
      arrayObj[i]=i
}
```

### 2．数组元素的输出

当把数组对象中的元素值输出时，有 3 种方法，分别介绍如下。

1）用下标获取指定元素值

该方法通过数组对象的下标获取指定的元素值。例如，获取数组对象中第 3 个元素的值，代码如下。

```
arrayObj = new Array("a","b","c","d")
var s=arrayObj[2]
```

运行结果如下：

```
c
```

2）用 for 语句获取数组中的元素值

该方法利用 for 语句获取数组对象中所有元素的值。例如：

```
arrayObj = new Array("a","b","c","d")
for(var i=0;i<arrayObj.length;i++){
      str=str+arrayObj[i].toString();
}
document.write(str);
```

运行结果如下：

```
abcd
```

3）用数组对象名输出所有元素值

该方法用创建的数组对象显示数组中的所有元素值。例如：

```
arrayObj = new Array("a","b","c","d")
document.write(arrayObj);
```

运行结果如下：

```
Abcd
```

**【例 5.8】**输出 3 名学霸姓名。**（实例位置：资源包\TM\sl\5\08）**

某班级里有 3 名学霸，创建一个存储 3 名学霸姓名（张三、李四、王五）的数组，然后输出这 3 个数组元素。首先创建一个包含 3 个元素的数组，并为每个数组元素赋值，然后使用 for 循环语句遍历输出数组中的所有元素。本例代码如下：

```
<script type="text/javascript">
var students = new Array(3);                    //定义数组
students[0] = "张三";                           //为下标为 0 的数组元素赋值
students[1] = "李四";                           //为下标为 1 的数组元素赋值
students[2] = "王五";                           //为下标为 2 的数组元素赋值
```

```
for(var i=0;i<3;i++){
    document.write("第"+(i+1)+"个学霸姓名是："+students[i]+"<br>");       //循环输出数组元素
}
</script>
```

运行结果如图 5.10 所示。

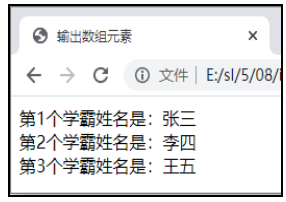



图 5.10　使用数组存储学霸姓名

**误区警示**

输出的数组元素的下标不能超过数组的范围。例如，下列代码会输出 undefined。

```
var arr= new Array("a","b");      //定义包含两个元素的数组
document.write(arr[2]);           //输出下标为 2 的元素的值
```

## 5.3.3　数组对象的属性

在数组对象中有 length 和 prototype 两个属性，下面分别对其进行介绍。

### 1. length 属性

length 属性用于返回数组的长度，其语法格式如下：

```
array.length
```

☑　array：表示数组名称。

例如，获取已创建的数组对象的长度，代码如下。

```
var arr=new Array(1,2,3,4,5,6,7,8);
document.write(arr.length);
```

运行结果如下：

```
8
```

例如，增加已有数组的长度，代码如下。

```
var arr=new Array(1,2,3,4,5,6,7,8);
arr[arr.length]=arr.length+1;
document.write(arr.length);
```

运行结果如下：

```
9
```

**注意**

当使用 new Array()方法创建数组时，并不对其进行赋值，length 属性的返回值为 0。

**【例 5.9】**输出省份、省会以及旅游景点。（**实例位置：资源包\TM\sl\5\09**）

将东北三省的省份名称、省会城市名称以及 3 个城市的旅游景点分别定义在数组中，应用 for 循环

语句和数组的 length 属性，将省份、省会以及旅游景点循环输出在表格中。本例代码如下。

```
<table cellspacing="1" bgcolor="#CC00FF">
  <tr height="30" bgcolor="#FFFFFF">
    <td align="center" width="50">序号</td>
    <td align="center" width="100">省份</td>
    <td align="center" width="100">省会</td>
    <td align="center" width="260">旅游景点</td>
  </tr>
<script type="text/javascript">
var province=new Array("黑龙江省","吉林省","辽宁省");                    //定义省份数组
var city=new Array("哈尔滨市","长春市","沈阳市");                        //定义省会数组
var tourist=new Array("太阳岛  圣索菲亚教堂  中央大街","净月潭  长影世纪城  动植物公园",
                      "沈阳故宫  沈阳北陵  张氏帅府");                   //定义旅游景点数组
for(var i=0; i<province.length; i++){                                  //定义 for 循环语句
    document.write("<tr height=26 bgcolor='#FFFFFF'>");                //输出<tr>开始标记
    document.write("<td align='center'>"+(i+1)+"</td>");               //输出序号
    document.write("<td align='center'>"+province[i]+"</td>");         //输出省份名称
    document.write("<td align='center'>"+city[i]+"</td>");             //输出省会名称
    document.write("<td align='center'>"+tourist[i]+"</td>");          //输出旅游景点
    document.write("</tr>");                                           //输出</tr>结束标记
}
</script>
</table>
```

运行结果如图 5.11 所示。

### 2. prototype 属性

prototype 属性的语法与 Object 对象的 prototype 属性相同。下面以例子的形式对该属性的应用进行说明。

**【例 5.10】**应用自定义方法输出数组。（**实例位置：资源包\TM\sl\5\10**）

应用数组对象的 prototype 属性自定义一个方法，用于显示数组中的全部数据。本例代码如下。

```
<h3>输出数组中所有数组元素</h3>
<script type="text/javascript">
Array.prototype.outAll=function(ar){                                   //自定义 outAll()方法
    for(var i=0;i<this.length;i++){                                    //定义 for 循环语句
        document.write(this[i]);                                       //输出数组元素
        document.write(ar);                                            //输出数组元素之间的分隔符
    }
}
var arr=new Array(1,2,3,4,5,6,7,8);                                    //定义数组
arr.outAll(" ");                                                       //调用自定义的 outAll()方法
</script>
```

运行结果如图 5.12 所示。


图 5.11　输出省份、省会和旅游景点


图 5.12　应用自定义方法输出数组中的所有数组元素

## 5.3.4　数组对象的方法

数组对象中的方法及其说明如表 5.5 所示。

表 5.5　数组对象的方法及其说明

| 方　法 | 说　明 |
|---|---|
| concat() | 连接两个或更多的数组，并返回结果 |
| pop() | 删除并返回数组中的最后一个元素 |
| push() | 向数组的尾部添加一个或多个元素，并返回新的长度 |
| shift() | 删除并返回数组中的第一个元素 |
| splice() | 删除元素，并向数组中添加新元素 |
| unshift() | 向数组的头部添加一个或多个元素，并返回新的长度 |
| reverse() | 颠倒数组中元素的顺序 |
| sort() | 对数组中的元素进行排序 |
| slice() | 从某个已有的数组中返回选定的元素 |
| toSource() | 代表对象的源代码 |
| toString() | 把数组转换为字符串，并返回结果 |
| toLocaleString() | 把数组转换为本地字符串，并返回结果 |
| join() | 把数组中的所有元素放入一个字符串，元素间通过指定的分隔符进行分隔 |
| valueOf() | 返回数组对象的原始值 |

## 5.3.5　数组的添加和删除

数组的添加和删除可以使用 concat()、shift()、pop()、push()和 unshift()方法实现。

### 1．concat()方法

concat()方法用于将其他数组连接到当前数组的尾端。concat()方法语法格式如下：

```
arrayObject.concat(arrayX,arrayX,...,arrayX)
```

☑　arrayObject：必选项。表示数组名称。

☑ arrayX：必选项。该参数可以是具体的值，也可以是数组对象。

例如，在数组的尾部添加数组元素，代码如下。

```
var arr=new Array(1,2,3,4,5,6,7,8);
document.write(arr.concat(9,10));
```

运行结果如图 5.13 所示。

例如，在数组的尾部添加其他数组，代码如下。

```
var arr1=new Array('a','b','c');
var arr2=new Array('d','e','f');
document.write(arr1.concat(arr2));
```

运行结果如图 5.14 所示。

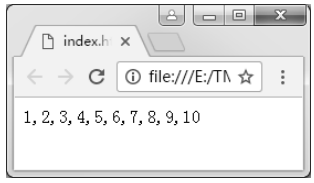



图 5.13 向数组中添加元素

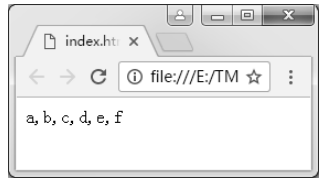



图 5.14 向数组中添加数组

### 2．shift()方法

shift()方法用于把数组中的第一个元素从数组中删除，并返回删除元素的值。shift()方法的语法格式如下：

```
arrayObject.shift()
```

☑ arrayObject：必选项。表示数组名称。

☑ 返回值：在数组中删除的第一个元素的值。

例如，删除数组中的第一个元素，代码如下。

```
var arr=new Array(1,2,3,4,5,6,7,8);
var Del=arr. shift();
document.write('删除元素为:'+Del+';删除后的数组为:'+arr);
```

运行结果如图 5.15 所示。

### 3．pop()方法

pop()方法用于删除并返回数组中的最后一个元素。pop()方法的语法格式如下：

```
arrayObject.pop()
```

☑ 返回值：数组对象的最后一个元素。

例如，删除数组中的最后一个元素，代码如下。

```
var arr=new Array(1,2,3,4,5,6,7,8);
var Del=arr.pop();
document.write('删除元素为:'+Del+';删除后的数组为:'+arr);
```

运行结果如图 5.16 所示。

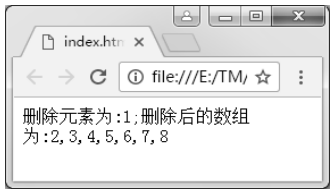



图 5.15　删除数组中的第一个元素

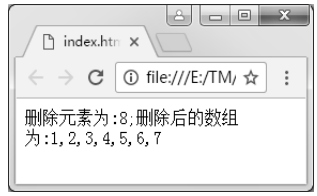



图 5.16　删除数组中的最后一个元素

### 4．push()方法

push()方法用于向数组的末尾添加一个或多个元素，并返回添加后的数组长度。push()方法的语法格式如下：

```
arrayObject.push(newelement1,newelement2,...,newelementX)
```

push()方法中的参数及其说明如表 5.6 所示。

表 5.6　push()方法中的参数及其说明

| 参　数 | 说　明 |
|---|---|
| arrayObject | 必选项，表示数组名称 |
| newelement1 | 必选项，表示要添加到数组中的第一个元素 |
| newelement2 | 可选项，表示要添加到数组中的第二个元素 |
| newelementX | 可选项，表示要添加到数组中的第 X 个元素 |

☑　返回值为把指定的值添加到数组中后的新长度。

例如，向数组的尾部添加元素 5、6 和 7，代码如下。

```
var arr=new Array(1,2,3,4);
document.write('原数组:'+arr+'<br>');
document.write('添加元素后的数组长度:'+arr.push(5,6,7)+'<br>');
document.write('新数组:'+arr);
```

运行结果如图 5.17 所示。

### 5．unshift()方法

unshift()方法向数组的头部添加一个或多个元素。unshift()方法的语法格式如下：

```
arrayObject.unshift(newelement1,newelement2,...,newelementX)
```

unshift()方法中的参数及其说明如表 5.7 所示。

表 5.7　unshift()方法中的参数及其说明

| 参　数 | 说　明 |
|---|---|
| arrayObject | 必选项，表示数组名称 |
| newelement1 | 必选项，表示向数组中添加的第一个元素 |
| newelement2 | 可选项，表示向数组中添加的第二个元素 |
| newelementX | 可选项，表示向数组中添加的第 X 个元素 |

例如，向 arr 数组的头部添加元素 1、2 和 3，代码如下。

```
var arr=new Array(4,5,6,7);
document.write('原数组:'+arr+'<br\>');
arr.unshift(1,2,3);
document.write('新数组:'+arr);
```

运行程序，会将原数组和新数组中的内容显示在页面中，如图 5.18 所示。

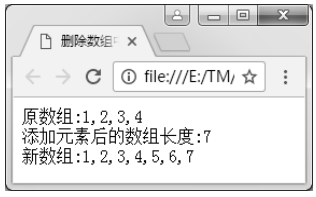



图 5.17　向数组的尾部添加元素

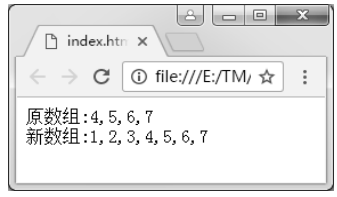



图 5.18　向数组的头部添加元素

## 5.3.6　设置数组的排列顺序

通过 reverse()和 sort()方法可以将数组中的元素按照指定的顺序进行排列。

### 1．reverse()方法

reverse()方法用于颠倒数组中元素的顺序。该方法的语法格式如下：

```
arrayObject.reverse()
```

☑　arrayObject：必选项，表示数组名称。

**注意**

reverse()方法会改变原来的数组，但不创建新数组。

例如，将数组中的元素顺序颠倒后显示，代码如下。

```
var arr=new Array(1,2,3,4);
document.write('原数组:'+arr+'<br\>');
arr.reverse();
document.write('颠倒后的数组:'+arr);
```

运行结果如图 5.19 所示。

### 2．sort()方法

sort()方法用于对数组的元素进行排序。该方法的语法格式如下：

```
arrayObject.sort(sortby)
```

☑　arrayObject：必选项，表示数组名称。

☑　sortby：可选项，规定排序的顺序，必须是函数。

**说明**

如果调用sort()方法时没有使用参数，将按字母顺序对数组中的元素进行排序，也就是按照字符的编码顺序排序。如果想按照其他标准排序，就需要提供比较函数。

例如，将数组中的元素按字符的编码顺序显示，代码如下。

```
var arr=new Array(2,1,4,3);
document.write('原数组:'+arr+'<br\>');
arr.sort();
document.write('排序后的数组:'+arr);
```

运行上述代码，将原数组和排序后的数组输出，结果如图 5.20 所示。


图 5.19　将数组颠倒输出

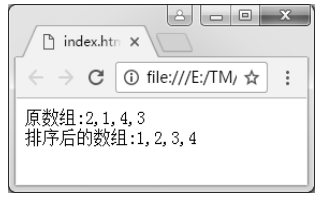



图 5.20　输出排序前与排序后的数组元素

如果想要将数组元素按照其他方法进行排序，就需要指定 sort()方法的参数。该参数通常是一个比较函数，该函数应该有两个参数（假设为 a 和 b）。在对元素进行排序时，每次比较两个元素都会执行比较函数，并将这两个元素作为参数传递给比较函数。其返回值有以下两种情况：

☑　如果返回值大于 0，则交换两个元素的位置。

☑　如果返回值小于或等于 0，则不进行任何操作。

例如，定义一个包含 4 个元素的数组，将数组中的元素按从小到大的顺序输出，代码如下。

```
var arr=new Array(9,6,10,5);                    //定义数组
document.write('原数组：'+arr+'<br>');           //输出原数组
function ascOrder(x,y){                         //定义比较函数
    if(x>y){                                    //如果第一个参数值大于第二个参数值
        return 1;                               //返回 1
    }else{
        return -1;                              //返回-1
    }
}
arr.sort(ascOrder);                             //对数组进行排序
document.write('排序后的数组：'+arr);            //输出排序后的数组
```

运行结果如图 5.21 所示。

**【例 5.11】**输出 2016 年电影票房排行榜前五名。**（实例位置：资源包\TM\sl\5\11）**

将 2016 年电影票房排行榜前五名的影片名称和对应的影片票房分别定义在数组中，对影片票房进行降序排序，将排序后的影片排名、影片名称和票房输出在表格中。本例代码如下。

```
<table cellspacing="1" bgcolor="#CC00FF">
  <tr height="30" bgcolor="#FFFFFF">
    <td align="center" width="50">排名</td>
```

```
    <td align="center" width="210">影片</td>
    <td align="center" width="100">票房</td>
  </tr>
<script type="text/javascript">
//定义影片数组 movieArr
var movieArr=new Array("魔兽","美人鱼","西游记之孙悟空三打白骨精","疯狂动物城","美国队长 3");
var boxofficeArr=new Array(14.7,33.9,12,15.3,12.5);          //定义票房数组 boxofficeArr
var sortArr=new Array(14.7,33.9,12,15.3,12.5);               //定义票房数组 sortArr
function ascOrder(x,y){                                      //定义比较函数
    if(x<y){                                                 //如果第一个参数值小于第二个参数值
        return 1;                                            //返回 1
    }else{
        return -1;                                           //返回-1
    }
}
sortArr.sort(ascOrder);                                      //为票房进行降序排序
for(var i=0; i<sortArr.length; i++){                         //定义外层 for 循环语句
    for(var j=0; j<sortArr.length; j++){                     //定义内层 for 循环语句
        if(sortArr[i]==boxofficeArr[j]){                     //分别获取排序后的票房在原票房数组中的索引
            document.write("<tr height=26 bgcolor='#FFFFFF'>");          //输出<tr>标记
            document.write("<td align='center'>"+(i+1)+"</td>");         //输出影片排名
            //输出票房对应的影片名称
            document.write("<td class='left'>"+movieArr[j]+"</td>");
            document.write("<td align='center'>"+sortArr[i]+"亿元</td>");  //输出票房
            document.write("</tr>");                                     //输出</tr>标记
        }
    }
}
</script>
</table>
```

运行结果如图 5.22 所示。


图 5.21　输出排序前与排序后的数组元素


| 排名 | 影片 | 票房 |
|---|---|---|
| 1 | 美人鱼 | 33.9亿元 |
| 2 | 疯狂动物城 | 15.3亿元 |
| 3 | 魔兽 | 14.7亿元 |
| 4 | 美国队长3 | 12.5亿元 |
| 5 | 西游记之孙悟空三打白骨精 | 12亿元 |

图 5.22　输出 2016 年电影票房排行榜前五名

## 5.3.7　获取数组中的某段数组元素

获取数组中的某段数组元素主要用 slice()方法实现。

slice()方法可从已有的数组中返回选定的元素。slice()方法的语法格式如下：

```
arrayObject.slice(start,end)
```

☑ start：必选项，规定从何处开始选取。如果是负数，说明从数组尾部开始选取。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。

☑ end：可选项，规定从何处结束选取。该参数是数组片段结束处的数组下标。如果没有指定该参数，那么截取的数组包含从 start 位置开始到数组结束的所有元素；如果这个参数是负数，那么将从数组尾部开始算起。

☑ 返回值：返回截取后的数组元素，该方法返回的数据中不包括 end 索引所对应的数据。

例如，获取指定数组中某段数组元素，代码如下。

```
var arr=new Array("a","b","c","d","e","f");                          //定义数组
document.write("原数组："+arr+"<br>");                                 //输出原数组
//输出截取后的数组
document.write("获取数组中第 3 个元素后的所有元素："+arr.slice(2)+"<br>");
document.write("获取数组中第 2 个到第 5 个元素："+arr.slice(1,5)+"<br>");   //输出截取后的数组
document.write("获取数组中倒数第 2 个元素后的所有元素："+arr.slice(-2));     //输出截取后的数组
```

运行程序，会将原数组以及截取数组中元素后的数据输出，运行结果如图 5.23 所示。

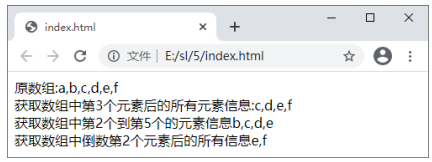



图 5.23　获取数组中某段数组元素

**【例 5.12】**计算歌手的最终得分。**（实例位置：资源包\TM\sl\5\12）**

某歌手参加歌唱比赛，5 位评委分别给出的分数是 95、90、89、91、96，要计算该歌手的最终得分，需要去掉一个最高分和一个最低分，并计算剩余 3 个分数的平均分，代码如下。

```
<script type="text/javascript">
var scoreArr=new Array(95,90,89,91,96);                              //定义分数数组
var scoreStr="";                                                     //定义分数字符串变量
for(var i=0; i<scoreArr.length; i++){
    scoreStr+=scoreArr[i]+"分  ";                                    //对所有分数进行连接
}
function ascOrder(x,y){                                              //定义比较函数
    if(x<y){                                                         //如果第一个参数值小于第二个参数值
        return 1;                                                    //返回 1
    }else{
        return -1;                                                   //返回-1
    }
}
scoreArr.sort(ascOrder);                                             //为分数进行降序排序
var newArr=scoreArr.slice(1,scoreArr.length-1);                      //去除最高分和最低分
var totalScore=0;                                                    //定义总分变量
for(var i=0; i<newArr.length; i++){
    totalScore+=newArr[i];                                           //计算总分
}
document.write("5 位评委打分："+scoreStr);                            //输出 5 位评委的打分
document.write("<br>去掉一个最高分："+scoreArr[0]+"分");              //输出去掉的最高分
//输出去掉的最低分
```

```
document.write("<br>去掉一个最低分："+scoreArr[scoreArr.length-1]+"分");
document.write("<br>歌手最终得分："+totalScore/newArr.length+"分");    //输出歌手最终得分
</script>
```

运行程序，结果如图 5.24 所示。

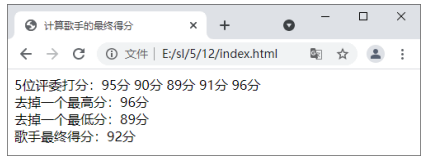



图 5.24　计算歌手的最终得分

## 5.3.8　将数组转换成字符串

将数组转换成字符串主要通过 toString()、toLocaleString()和 join()方法实现。

### 1．toString()方法

toString()方法可把数组转换成字符串，并返回结果。toString()方法的语法格式如下：

```
arrayObject.toString()
```

☑　arrayObject：必选项，表示数组名称。
☑　返回值：以字符串显示 arrayObject。返回值与没有参数的 join()方法返回的字符串相同。

**注意**

转换成字符串后，数组中的各个元素之间以逗号进行分隔。

例如，将数组转换成字符串，代码如下。

```
var arr=new Array("a","b","c","d","e","f");
document.write(arr.toString());
```

运行结果如图 5.25 所示。

### 2．toLocaleString()方法

toLocaleString()方法用于将数组转换成本地字符串。toLocaleString()方法的语法格式如下：

```
arrayObject.toLocaleString()
```

☑　arrayObject：必选项，表示数组名称。
☑　返回值：本地字符串。

**说明**

首先调用每个数组元素的 toLocaleString()方法，然后使用地区特定的分隔符把生成的字符串连接起来，形成一个字符串。

例如，将数组转换为本地字符串，代码如下。

```
var arr=new Array("a","b","c","d","e","f");
document.write(arr.toLocaleString());
```

运行结果与图 5.25 一样。

### 3．join()方法

join()方法用于将数组中的所有元素放入一个字符串中。join()方法的语法格式如下。

```
arrayObject.join(separator)
```

参数说明：

☑ separator：可选项，表示要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

☑ 返回值：返回一个字符串。该字符串是把 arrayObject 的每个元素转换为字符串，然后把这些字符串连接起来，在两个元素之间插入 separator 字符串而生成的。

例如，以指定的分隔符将数组中的元素转换成字符串，代码如下。

```
var arr=new Array("a","b","c","d","e","f");
document.write(arr.join("#"));
```

运行结果如图 5.26 所示。

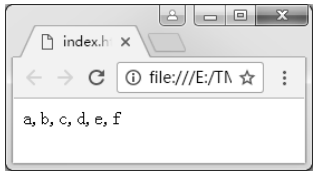



图 5.25　将数组转换为字符串


图 5.26　使用指定分隔符将数组中的元素转换为字符串

**编程训练（答案位置：资源包\TM\sl\5\编程训练\）**

【训练 4】获取当前日期和星期　为了使整个日期信息显示得更详细一些，一般都会在日期后面显示一个星期。通过数组来显示当前的星期。

【训练 5】获取元素的最大值　利用数组中的 prototype 属性自定义一个用于获取元素中最大值的方法，找到 1、3、6、2、9、5 这几个数字中的最大值。

【训练 6】获取数组中的指定元素　获取数组（1,2,3,4,5,6）中第 2 个到倒数第 2 个数组元素。（提示：使用数组对象中的 slice()方法）

## 5.4　实践与练习

**答案位置：（资源包\TM\sl\5\实践与练习\）**

综合练习 1：计算从出生到现在度过的时间　假设周星星的出生日期是 1996 年 6 月 9 日，计算周星星从出生到现在已经度过了多长时间，要求计算出天数、小时数、分钟数和秒数。

综合练习 2：输出列出的途经站　由长春站始发，到北京站的 D20 次列车途经车站为昌图西、铁

岭西、沈阳北和绥中北。编写程序，输出该次列车的途经站，以及反向输出该车次的途经站。实现效果如下。

```
途经站：长春 昌图西 铁岭西 沈阳北 绥中北 北京
反向站：北京 绥中北 沈阳北 铁岭西 昌图西 长春
```

综合练习 3：2018 年国内手机销量排行榜　将 2018 年国内手机销量排行榜前九名的手机品牌和总销量定义在数组中，按手机销量进行升序排序后，将手机销量排名、手机品牌和总销量输出在页面中。实现效果如图 5.27 所示。

| 排名 | 品牌 | 总销量（万部） |
|---|---|---|
| 9 | Gionee | 478 |
| 8 | Samsung | 595 |
| 7 | Meizu | 948 |
| 6 | MI | 4796 |
| 5 | iPhone | 5270 |
| 4 | HONOR | 5427 |
| 3 | HUAWEI | 6490 |
| 2 | vivo | 7464 |
| 1 | OPPO | 7637 |

图 5.27　手机销量升序排列