**第8章 对象和数组**

学习要点：

1. Object类型
2. Array类型
3. 对象中的方法

什么是对象，其实就是一种类型，即引用类型。而对象的值就是引用类型的实例。在ECMAScript中引用类型是一种数据结构，用于将数据和功能组织在一起。它也常被称做为类，但ECMAScript中却没有这种东西。虽然ECMAScript是一门面向对象的语言，却不具备传统面向对象语言所支持的类和接口等基本结构。

1. **Object类型**

到目前为止，我们使用的引用类型最多的可能就是Object类型了。虽然Object的实例不具备多少功能，但对于在应用程序中的存储和传输数据而言，它确实是非常理想的选择。

创建Object类型有两种。一种是使用new运算符，一种是字面量表示法。

1.使用new运算符创建Object

var box = new Object(); //new方式

box.name = 'jack'; //创建属性字段

box.age = 28; //创建属性字段

1. new关键字可以省略

var box = Object(); //省略了new关键字

3.使用字面量方式创建Object

var box = { //字面量方式

name : 'jack', //创建属性字段

age : 28

};

4.属性字段也可以使用字符串形式

var box = {

'name' : 'jack', //也可以用字符串形式

'age' : 28

};

5.使用字面量及传统复制方式

var box = {}; //字面量方式声明空的对象

box.name = 'jack'; //点符号给属性复制

box.age = 28;

6.两种属性输出方式

alert(box.age); //点表示法输出

alert(box['age']); //中括号表示法输出，注意引号

PS：在使用字面量声明Object对象时，不会调用Object()构造函数(Firefox除外)。

7.给对象创建方法

var box = {

run : function () { //对象中的方法

return '运行';

}

}

alert(box.run()); //调用对象中的方法

8.使用delete删除对象属性

delete box.name; //删除属性

在实际开发过程中，一般我们更加喜欢字面量的声明方式。因为它清晰，语法代码少，而且还给人一种封装的感觉。字面量也是向函数传递大量可选参数的首选方式。

function box(obj) { //参数是一个对象

if (obj.name != undefined) alert(obj.name); //判断属性是否存在

if (obj.age != undefined) alert(obj.age);

}

box({ //调用函数传递一个对象

name : 'jack',

age : 28

});

1. **Array类型**

除了Object类型之外，Array类型是ECMAScript最常用的类型。而且ECMAScript中的Array类型和其他语言中的数组有着很大的区别。虽然数组都是有序排列，但ECMAScript中的数组每个元素可以保存任何类型。ECMAScript中数组的大小也是可以调整的。

创建Array类型有两种方式：第一种是new运算符，第二种是字面量。

1.使用new关键字创建数组

var box = new Array(); //创建了一个数组

var box = new Array(10); //创建一个包含10个元素的数组

var box = new Array('jack',28,'教师','盐城'); //创建一个数组并分配好了元素

2.以上三种方法，可以省略new关键字。

var box = Array(); //省略了new关键字

3使用字面量方式创建数组

var box = []; //创建一个空的数组

var box = ['jack',28,'教师','盐城']; //创建包含元素的数组

var box = [1,2,]; //禁止这么做，IE会识别3个元素

var box = [,,,,,]; //同样，IE的会有识别问题

PS：和Object一样，字面量的写法不会调用Array()构造函数。(Firefox除外)。

1. 使用索引下标来读取数组的值

alert(box[2]); //获取第三个元素

box[2] = '学生'; //修改第三个元素

box[4] = '计算机编程'; //增加第五个元素

1. 使用length属性获取数组元素量

alert(box.length) //获取元素个数

box.length = 10; //强制元素个数

box[box.length] = 'JS技术'; //通过length给数组增加一个元素

1. 创建一个稍微复杂一点的数组

var box = [

{ //第一个元素是一个对象

name : 'jack',

age : 28,

run : function () {

return 'run了';

}

},

['马云','李彦宏',new Object()],//第二个元素是数组

'江苏', //第三个元素是字符串

25+25, //第四个元素是数值

new Array(1,2,3) //第五个元素是数组

];

alert(box);

PS：数组最多可包含4294967295个元素，超出即会发生异常。

1. **对象中的方法**

**转换方法**

对象或数组都具有toLocaleString()、toString()和valueOf()方法。其中toString()和valueOf()无论重写了谁，都会返回相同的值。数组会讲每个值进行字符串形式的拼接，以逗号隔开。

var box = ['jack',28,'计算机编程']; //字面量数组

alert(box); //隐式调用了toString()

alert(box.toString()); //和valueOf()返回一致

alert(box.toLocaleString()); //返回值和上面两种一致

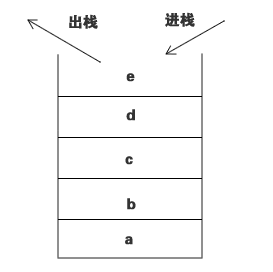
默认情况下，数组字符串都会以逗号隔开。如果使用join()方法，则可以使用不同的分隔符来构建这个字符串。

var box = ['jack', 28, '计算机编程'];

alert(box.join('|')); //李炎恢|28|计算机编程

**栈方法**

ECMAScript数组提供了一种让数组的行为类似于其他数据结构的方法。也就是说，可以让数组像栈一样，可以限制插入和删除项的数据结构。栈是一种数据结构(后进先出)，也就是说最新添加的元素最早被移除。而栈中元素的插入(或叫推入)和移除(或叫弹出)，只发生在一个位置——栈的顶部。ECMAScript为数组专门提供了push()和pop()方法。



push()方法可以接收任意数量的参数，把它们逐个添加到数组的末尾，并返回修改后数组的长度。而pop()方法则从数组末尾移除最后一个元素，减少数组的length值，然后返回移除的元素。

var box = ['jack', 28, '计算机编程']; //字面量声明

alert(box.push('盐城')); //数组末尾添加一个元素，并且返回长度

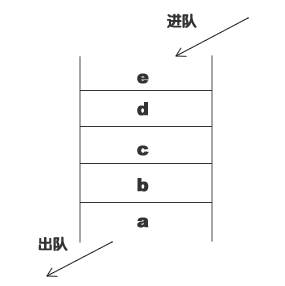
alert(box); //查看数组

box.pop(); //移除数组末尾元素，并返回移除的元素

alert(box); //查看元素

**队列方法**

栈方法是后进先出，而列队方法就是先进先出。列队在数组的末端添加元素，从数组的前端移除元素。通过push()向数组末端添加一个元素，然后通过shift()方法从数组前端移除一个元素。



var box = ['jack', 28, '计算机编程']; //字面量声明

alert(box.push('盐城')); //数组末尾添加一个元素，并且返回长度

alert(box); //查看数组

alert(box.shift()); //移除数组开头元素，并返回移除的元素

alert(box); //查看数组

ECMAScript还为数组提供了一个unshift()方法，它和shift()方法的功能完全相反。unshift()方法为数组的前端添加一个元素。

var box = ['jack', 28, '计算机编程']; //字面量声明

alert(box.unshift('盐城','江苏')); //数组开头添加两个元素

alert(box); //查看数组

alert(box.pop()); //移除数组末尾元素，并返回移除的元素

alert(box); //查看数组

PS：IE浏览器对unshift()方法总是返回undefined而不是数组的新长度。

**重排序方法**

数组中已经存在两个可以直接用来排序的方法：reverse()和sort()。

reverse() 逆向排序

var box = [1,2,3,4,5]; //数组

alert(box.reverse()); //逆向排序方法，返回排序后的数组

alert(box); //源数组也被逆向排序了，说明是引用

sort() 从小到大排序

var box = [4,1,7,3,9,2]; //数组

alert(box.sort()); //从小到大排序，返回排序后的数组

alert(box); //源数组也被从小到大排序了

sort方法的默认排序在数字排序上有些问题，因为数字排序和数字字符串排序的算法是一样的。我们必须修改这一特征，修改的方式，就是给sort(参数)方法传递一个函数参数。这点可以参考手册说明。

function compare(value1, value2) { //数字排序的函数参数

if (value1 < value2) { //小于，返回负数

return -1;

} else if (value1 > value2) { //大于，返回正数

return 1;

} else { //其他，返回0

return 0;

}

var box = [0,1,5,10,15]; //验证数字字符串，和数字的区别

alert(box.sort(compare)); //传参

PS：如果要反向操作，即从大到小排序，正负颠倒即可。当然，如果要逆序用reverse()更加方便。

**操作方法**

ECMAScript为操作已经包含在数组中的元素提供了很多方法。concat()方法可以基于当前数组创建一个新数组。slice()方法可以基于当前数组获取指定区域元素并创建一个新数组。splice()主要用途是向数组的中部插入元素。

var box = ['jack', 28, '盐城']; //当前数组

var box2 = box.concat('计算机编程'); //创建新数组，并添加新元素

alert(box2); //输出新数组

alert(box); //当前数组没有任何变化

var box = ['jack', 28, '盐城']; //当前数组

var box2 = box.slice(1); //box.slice(1,3)，2-4之间的元素

alert(box2); //28，盐城

alert(box); //当前数组

splice中的删除功能：

var box = ['jack', 28, '盐城']; //当前数组

var box2 = box.splice(0,2); //截取前两个元素

alert(box2); //返回截取的元素

alert(box); //当前数组被截取的元素被删除

splice中的插入功能：

var box = ['jack', 28, '盐城']; //当前数组

var box2 = box.splice(1,0,'计算机编程','江苏'); //没有截取，但插入了两条

alert(box2); //在第2个位置插入两条

alert(box); //输出

splice中的替换功能：

var box = ['jack', 28, '盐城']; //当前数组

var box2 = box.splice(1,1,100); //截取了第2条，替换成100

alert(box2); //输出截取的28

alert(box); //输出数组