17.6 赋值运算符

赋值是一种运算，但习惯上，把赋值独立成行，故称之为赋值语句。

```
var a,b;                    // 定义变量
a = null;                   // 给变量赋值
b = undefined;              // 给变量赋值
```

赋值运算符的左侧运算数必须是变量、对象属性或数组元素。

【示例 1】下面的写法是不对的，因为左侧的值是一个直接量，是不允许操作的。

```
1 = 100;                    // 返回错误
```

JavaScript 提供了两种类型的赋值运算符：简单赋值运算符（=）和附加操作的赋值运算符。

简单的赋值运算符，就是把右侧的运算数的值直接复制给左侧变量。

附加操作的赋值运算符，就是赋值之前还要对右侧运算数执行某种操作，然后再复制，详细说明如表 E17.11 所示。

表 E17.11 附加操作赋值运算符

赋值运算符	说明	示例	转化
+=	加法运算或连接操作并赋值	a += b	a = a + b
-=	减法运算并赋值	a -= b	a = a - b
*=	乘法运算并赋值	a *= b	a = a * b
/=	除法运算并赋值	a /= b	a = a / b
%=	取模运算并赋值	a %= b	a = a % b
<<=	左移位运算并赋值	a <<= b	a = a << b
>>=	右移位运算并赋值	a >>= b	a = a >> b
>>>=	无符号右移位运算并赋值	a >>>= b	a = a >>> b
&=	位与运算并赋值	a &= b	a = a & b
|=	位或运算并赋值	a |= b	a = a | b
^=	位异或运算并赋值	a ^= b	a = a ^ b

【示例 2】由于赋值运算符可以参与表达式运算，用户可以设计很多复杂的赋值操作，如连续赋值表达式。

```
var a = b = c = d = e = f = 100;        // 连接赋值
```

由于赋值运算符是从右向左进行计算，所以连续赋值运算并不会发生错误。

在条件表达式中进行赋值：

```
for(var a = 1, b = 10; a < b; a ++ ){       // 在条件表达式中进行赋值操作
   alert(a);
}
```

【示例 3】在下面这个复杂的表达式中，逻辑与左侧的运算数是一个赋值表达式，右侧的运算数也是一个赋值表达式。但是左侧仅是一个简单的数值赋值，而右侧的是把一个函数对象赋值给了变量 b。在逻辑与运算中，左侧的赋值并没有真正的复制给变量 a，当逻辑与运算执行右侧的表达式时，该表达式是把一个函数赋值给变量 b，然后利用小括号运算符调用这个函数，返回变量 a 的值，结果并没有返回变量 a 的值为 6，而是 undefined。

```
var a;                              // 定义变量 a
alert(a = 6 && (b = function(){     // 逻辑与运算表达式
   return a;                        // 返回变量 a 的值
})()
);                                  // 结果返回 undefined
```

由于赋值运算作为表达式使用具有副作用，即它能够改变变量的值。因此在使用时要慎重，确保不要引发潜在的危险。经过上面示例代码，可以看到赋值运算符参与表达式运算时给变量 a 带来了不可预测的返回值。因此，对于上面表达式，更安全的写法是：

```
var a = 6;                          // 定义并初始化变量 a
b = function(){                     // 定义函数对象 b
   return a;
```

```
}
alert(a && b());                    // 利用逻辑与运算，根据 a 的逻辑值，决定是否调用函数 b
```