**delete**

delete 运算符删除对以前定义的对象属性或方法的引用。例如：

var o = new Object;

o.name = "David";

alert(o.name); //输出 "David"

delete o.name;

alert(o.name); //输出 "undefined"

在这个例子中，删除了 name 属性，意味着强制解除对它的引用，将其设置为 undefined（即创建的未初始化的变量的值）。

delete 运算符不能删除开发者未定义的属性和方法。例如，下面的代码将引发错误：

delete o.toString;

即使 toString 是有效的方法名，这行代码也会引发错误，因为 toString() 方法是原始的 ECMAScript 方法，不是开发者定义的。

**void**

void 运算符对任何值返回 undefined。该运算符通常用于避免输出不应该输出的值，例如，从 HTML 的 <a> 元素调用 JavaScript 函数时。要正确做到这一点，函数不能返回有效值，否则浏览器将清空页面，只显示函数的结果。例如：

<a href="javascript:window.open('about:blank')">Click me</a>

如果把这行代码放入 HTML 页面，点击其中的链接，即可看到屏幕上显示 "[object]"。TIY

这是因为 window.open() 方法返回了新打开的窗口的引用。然后该对象将被转换成要显示的字符串。

要避免这种效果，可以用 void 运算符调用 window.open() 函数：

<a href="javascript:void(window.open('about:blank'))">Click me</a>

这使 window.open() 调用返回 undefined，它不是有效值，不会显示在浏览器窗口中。

**提示：请记住，没有返回值的函数真正返回的都是 undefined。**