3.5.1 一元操作符

只能操作一个值的操作符叫做**一元操作符**。一元操作符是 ECMAScript 中最简单的操作符。

这个例子中变量 anotherAge 的初始值等于变量 age 的值前置递减之后加 2。由于先执行了减法操作，age 的值变成了 28，所以再加上 2 的结果就是 30。

由于前置递增和递减操作与执行语句的优先级相等，因此整个语句会从左至右被求值。再看一个例子：

```
var num1 = 2;
var num2 = 20;
var num3 = --num1 + num2;       // 等于 21
var num4 = num1 + num2;         // 等于 21
```

IncrementDecrementExample02.htm

在这里，num3 之所以等于 21 是因为 num1 先减去了 1 才与 num2 相加。而变量 num4 也等于 21 是因为相应的加法操作使用了 num1 减去 1 之后的值。

后置型递增和递减操作符的语法不变（仍然分别是++和--），只不过要放在变量的后面而不是前面。后置递增和递减与前置递增和递减有一个非常重要的区别，即递增和递减操作是在包含它们的语句被求值之后才执行的。这个区别在某些情况下不是什么问题，例如：

```
var age = 29;
age++;
```

把递增操作符放在变量后面并不会改变语句的结果，因为递增是这条语句的唯一操作。但是，当语句中还包含其他操作时，上述区别就会非常明显了。请看下面的例子：

```
var num1 = 2;
var num2 = 20;
var num3 = num1-- + num2;       // 等于 22
var num4 = num1 + num2;         // 等于 21
```

1. 递增和递减操作符

递增和递减操作符直接借鉴自 C，而且各有两个版本：前置型和后置型。顾名思义，前置型应该位

于要操作的变量之前，而后置型则应该位于要操作的变量之后。因此，在使用前置递增操作符给一个数

值加 1 时，要把两个加号（++）放在这个数值变量前面，如下所示：

```
var age = 29;
++age;
```

在这个例子中，前置递增操作符把 age 的值变成了 30（为 29 加上了 1）。实际上，执行这个前置递

增操作与执行以下操作的效果相同：

```
var age = 29;
age = age + 1;
```

执行前置递减操作的方法也类似，结果会从一个数值中减去 1。使用前置递减操作符时，要把两个

减号（ -- ）放在相应变量的前面，如下所示：

```
var age = 29;
--age;
```

这样， age 变量的值就减少为 28（从 29 中减去了 1）。

执行前置递增和递减操作时，变量的值都是在语句被求值以前改变的。（在计算机科学领域，这种

情况通常被称作副效应。）请看下面这个例子。

```
var age = 29;
var anotherAge = --age + 2;
alert(age); // 输出 28
alert(anotherAge); // 输出 30
```

后置型递增和递减操作符的语法不变（仍然分别是 ++ 和 -- ），只不过要放在变量的后面而不是前面。

后置递增和递减与前置递增和递减有一个非常重要的区别，即递增和递减操作是在包含它们的语句被求

值之后才执行的。这个区别在某些情况下不是什么问题，例如：

```
var age = 29;
age++;
```

这个例子中变量 anotherAge 的初始值等于变量 age 的值前置递减之后加 2。由于先执行了减法操作，age 的值变成了 28，所以再加上 2 的结果就是 30。

由于前置递增和递减操作与执行语句的优先级相等，因此整个语句会从左至右被求值。再看一个例子：

```
var num1 = 2;
var num2 = 20;
var num3 = --num1 + num2;       // 等于 21
var num4 = num1 + num2;         // 等于 21
```

IncrementDecrementExample02.htm

在这里，num3 之所以等于 21 是因为 num1 先减去了 1 才与 num2 相加。而变量 num4 也等于 21 是因为相应的加法操作使用了 num1 减去 1 之后的值。

后置型递增和递减操作符的语法不变（仍然分别是++和--），只不过要放在变量的后面而不是前面。后置递增和递减与前置递增和递减有一个非常重要的区别，即递增和递减操作是在包含它们的语句被求值之后才执行的。这个区别在某些情况下不是什么问题，例如：

```
var age = 29;
age++;
```

把递增操作符放在变量后面并不会改变语句的结果，因为递增是这条语句的唯一操作。但是，当语句中还包含其他操作时，上述区别就会非常明显了。请看下面的例子：

```
var num1 = 2;
var num2 = 20;
var num3 = num1-- + num2;       // 等于 22
var num4 = num1 + num2;         // 等于 21
```

所有这 4 个操作符对任何值都适用，也就是它们不仅适用于整数，还可以用于字符串、布尔值、浮点数值和对象。在应用于不同的值时，递增和递减操作符遵循下列规则。

- 在应用于一个包含有效数字字符的字符串时，先将其转换为数字值，再执行加减 1 的操作。字符串变量变成数值变量。
- 在应用于一个不包含有效数字字符的字符串时，将变量的值设置为 NaN（第 4 章将详细讨论）。字符串变量变成数值变量。
- 在应用于布尔值 false 时，先将其转换为 0 再执行加减 1 的操作。布尔值变量变成数值变量。
- 在应用于布尔值 true 时，先将其转换为 1 再执行加减 1 的操作。布尔值变量变成数值变量。
- 在应用于浮点数值时，执行加减 1 的操作。
- 在应用于对象时，先调用对象的 valueOf()方法（第 5 章将详细讨论）以取得一个可供操作的值。然后对该值应用前述规则。如果结果是 NaN，则在调用 toString()方法后再应用前述规则。对象变量变成数值变量。

2. 一元加和减操作符

一元加操作符以一个加号（+）表示，放在数值前面，对数值不会产生任何影响。

不过，在对非数值应用一元加操作符时，该操作符会像 Number() 转型函数一样对这个值执行转换。

在将一元减操作符应用于数值时，该值会变成负数（如上面的例子所示）。而当应用于非数值时，

一元减操作符遵循与一元加操作符相同的规则，最后再将得到的数值转换为负数