17.9.6 转换为数字

JavaScript 提供了两种静态方法把非数字的原始值转换为数字：parseInt()和 parseFloat()。其中 parseInt()可以把值转换为整数，而 parseFloat()可以把值转换为浮点数。

parseInt()和 parseFloat()函数对字符串类型的值有效，其他类型的值调用这两个函数都会返回 NaN。在转换字符串为数字之前，它们都会对字符串进行分析，以验证转换是否继续，具体分析如下。

1．使用 parseInt()

在开始转换时，parseInt()函数会先查看位置 0 处的字符，如果该位置不是有效数字，则将返回 NaN，不再深入分析。如果位置 0 处的字符是数字，则将查看位置 1 处的字符，并进行同样的测试，依此类推，在整个验证过程中，直到发现非数字字符为止，此时 parseInt()函数将把前面分析合法的数字字符转换为数值，并返回。例如：

```
alert(parseInt("123abc"));          // 返回数字 123
alert(parseInt("1.73"));            // 返回数字 1
alert(parseInt(".123"));            // 返回值 NaN
```

在浮点数中的点号对于 parseInt()函数来说是属于非法字符的，因此不会转换它，并返回。

如果以 0 为开头的数字字符串，则 parseInt()函数会把它作为八进制数字处理，先把它转换为数值，然后再转换为十进制的数字返回。如果以 0x 为开头的数字字符串，则 parseInt()函数会把它作为十六进制数字处理，先把它转换为数值，然后再转换为十进制的数字返回。

```
var d = "010";                      // 八进制数字字符串
var e = "0x10";                     // 十六进制数字字符串
alert(parseInt(d));                 // 返回十进制数字 8
alert(parseInt(e));                 // 返回十进制数字 16
```

parseInt()也支持基模式，可以把二进制、八进制、十六进制等不同进制的数字字符串转换为整数。基模式由 parseInt()函数的第二个参数指定。

【示例 1】下面代码把十六进制数字字符串"123abc"转换为十进制整数：

```
var a = "123abc";
alert(parseInt(a,16));        // 返回值十进制整数 1194684
```

【示例 2】下面代码把二进制、八进制和十进制数字字符串转换为整数：

```
alert(parseInt("10",2));            // 把二进制数字 10 转换为十进制整数为 2
alert(parseInt("10",8));            // 把八进制数字 10 转换为十进制整数为 8
alert(parseInt("10" ,10));          // 把十进制数字 10 转换为十进制整数为 10
```

【示例 3】如果第一个参数是十进制的值，包含 0 前缀，为了避免被误解为八进制的数字，则应该指定第二个参数值为 10，即显式定义基，而不是采用默认基。

```
alert(parseInt("010"));             // 把八进制数字 10 转换为十进制整数为 8
alert(parseInt("010",8));           // 把八进制数字 010 转换为十进制整数为 8
alert(parseInt("010",10));          // 把十进制数字 010 转换为十进制整数为 10
```

2．使用 parseFloat()函数

parseFloat()函数与 parseInt()函数用法基本相同。但是它能够识别第一个出现的小数点号，而第二个小数点号被视为非法的。

```
alert(parseFloat("1.2317.5"));      // 返回数值 1.234
```

此外，数字必须是十进制形式的字符串，而不能够使用八进制或十六进制的数字字符串。同时对于数字前面的 0（八进制数字标识）会忽略，而对于十六进制形式的数字，则返回 0 值。例如：

```
alert(parseFloat("123"));           // 返回数值 124
alert(parseFloat("123abc"));        // 返回数值 123
alert(parseFloat("010"));           // 返回数值 10
alert(parseFloat("0x10"));          // 返回数值 0
```

```
alert(parseFloat("x10"));          // 返回数值 NaN
```

3．使用乘号运算符

加号运算符不仅能够执行数值求和运算，还可以把字符串连接起来。由于 JavaScript 处理字符串连接操作的优先级要高于数字求和运算。因此，当数字字符串与数值使用加号连接时，将优先执行连接操作，而不是求和运算。例如：

```
var a = 1;                         // 数值
var b = "1";                       // 数字字符串
alert(a+b);                        // 返回字符串"11"
```

在执行表达式 a+b 的运算时，变量 a 先被转换为字符串，然后以求和进行计算，所以计算结果为字符串"11"，而不是数值 2。因此，我们常常使用加号运算符把一个值转换为字符串。

不过，如果让变量 b 乘以 1，则加号运算符就以求和进行计算了。例如：

```
var a = 1;                         // 数值
var b = "1";                       // 数字字符串
alert(a + (b * 1));                // 返回数值 2
```

如果让一个数字字符串变量乘以 1，则 JavaScript 解释器能够自动把数字字符串转换为数值，然后再继续求和运算，而不是进行字符串连接操作。