1. 加法

加法操作符（+）的用法如下所示：

```
var result = 1 + 2;
```

如果两个操作符都是数值，执行常规的加法计算，然后根据下列规则返回结果：

- 如果有一个操作数是 NaN，则结果是 NaN；
- 如果是 Infinity 加 Infinity，则结果是 Infinity；
- 如果是-Infinity 加-Infinity，则结果是-Infinity；
- 如果是 Infinity 加-Infinity，则结果是 NaN；
- 如果是+0 加+0，则结果是+0；
- 如果是–0 加–0，则结果是–0；
- 如果是+0 加–0，则结果是+0。

不过，如果有一个操作数是字符串，那么就要应用如下规则：

- 如果两个操作数都是字符串，则将第二个操作数与第一个操作数拼接起来；
- 如果只有一个操作数是字符串，则将另一个操作数转换为字符串，然后再将两个字符串拼接起来。

如果有一个操作数是对象、数值或布尔值，则调用它们的 toString()方法取得相应的字符串值，然后再应用前面关于字符串的规则。对于 undefined 和 null，则分别调用 String()函数并取得字符串"undefined"和"null"。

下面来举几个例子：

```
var result1 = 5 + 5;        // 两个数值相加
alert(result1);             // 10
```

```
var num1 = 5;
var num2 = 10;
var message = "The sum of 5 and 10 is " + num1 + num2;
alert(message);    // "The sum of 5 and 10 is 510"
```

AddExample02.htm

在这个例子中，变量 message 的值是执行两个加法操作之后的结果。有人可能以为最后得到的字符串是"The sum of 5 and 10 is 15"，但实际的结果却是"The sum of 5 and 10 is 510"。之所以会这样，是因为每个加法操作是独立执行的。第一个加法操作将一个字符串和一个数值（5）拼接了起来，结果是一个字符串。而第二个加法操作又用这个字符串去加另一个数值（10），当然也会得到一个字符串。如果想先对数值执行算术计算，然后再将结果与字符串拼接起来，应该像下面这样使用圆括号：

```
var num1 = 5;
var num2 = 10;
var message = "The sum of 5 and 10 is " + (num1 + num2);
alert(message);    //"The sum of 5 and 10 is 15"
```

2. 减法

减法操作符（–）是另一个极为常用的操作符，其用法如下所示：

```
var result = 2 - 1;
```

与加法操作符类似，ECMAScript 中的减法操作符在处理各种数据类型转换时，同样需要遵循一些特殊规则。如下所示：

□ 如果两个操作符都是数值，则执行常规的算术减法操作并返回结果；

□ 如果有一个操作数是 NaN ，则结果是 NaN ；

□ 如果是 Infinity 减 Infinity ，则结果是 NaN ；

□ 如果是 -Infinity 减 -Infinity ，则结果是 NaN ；

□ 如果是 Infinity 减 -Infinity ，则结果是 Infinity ；

□ 如果是 -Infinity 减 Infinity ，则结果是 -Infinity ；

□ 如果是+0 减+0，则结果是+0；

□ 如果是+0 减□0，则结果是□0；

□ 如果是□0 减□0，则结果是+0；

□ 如果有一个操作数是字符串、布尔值、 null 或 undefined ，则先在后台调用 Number() 函数将
其转换为数值，然后再根据前面的规则执行减法计算。如果转换的结果是 NaN ，则减法的结果
就是 NaN ；

□ 如果有一个操作数是对象，则调用对象的 valueOf() 方法以取得表示该对象的数值。如果得到
的值是 NaN ，则减法的结果就是 NaN 。如果对象没有 valueOf() 方法，则调用其 toString()
方法并将得到的字符串转换为数值。