# 1.JavaScript支持所有的算术操作如加减乘除,取余(modulo)%

## 案例:

|  |
| --- |
|  |

## 2.++和—操作符,表示自增和自减

# 扩展:js中number常见方法

**javascript中Number 类型 在实际开发中常用的一些操作方法**

在 JavaScript 中，Number 类型是非常基础的[数据类型](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B&spm=1001.2101.3001.7020)之一，用于表示整数和浮点数。除了基本的算术运算外，还有许多内置的方法可以帮助你处理数字。下面列举了一些在实际开发中常用的 Number 类型的操作方法：

**1. 转换方法**

**Number()**

将非数字值转换为数字：

Number('123') === 123; // true

Number(true); // 1

Number(false); // 0

**parseInt(), parseFloat()**

分别将字符串解析为整数或[浮点数](https://so.csdn.net/so/search?q=%E6%B5%AE%E7%82%B9%E6%95%B0&spm=1001.2101.3001.7020)：

parseInt('123.45'); // 123

parseFloat('123.45abc'); // 123.45

**isNaN()**

检查一个值是否为 NaN：

isNaN('abc'); // true

isNaN(123); // false

**isFinite()**

检查一个值是否为有限的数字：

isFinite(123); // true

isFinite(Infinity); // false

isFinite(NaN); // false

**2. 数学运算方法**

**Math 对象提供的方法**

* Math.abs(x)：返回 x 的绝对值。
* Math.ceil(x)：向上取整。
* Math.floor(x)：向下取整。
* Math.round(x)：四舍五入。
* Math.trunc(x)：去除小数部分，返回整数部分。
* Math.min(...values)：返回最小值。
* Math.max(...values)：返回最大值。
* Math.random()：返回一个介于 0（包括）和 1（不包括）之间的随机数。
* Math.sign(x)：返回 x 的符号（-1, 0, 1）。
* Math.pow(x, y)：返回 x 的 y 次幂。
* Math.sqrt(x)：返回 x 的平方根。
* Math.log(x)：返回 x 的自然对数。
* Math.exp(x)：返回 e 的 x 次幂。

**3. Number 原型方法**

**toFixed()**

将数字格式化为字符串，并保留指定位数的小数：

123.456.toFixed(2); // "123.46"

**toLocaleString()**

将数字转换为本地化的字符串：

123456789.toLocaleString('en-US'); // "123,456,789"

123456789.toLocaleString('zh-CN'); // "123,456,789" 或 "123,456,789"，取决于实现

**toPrecision()**

将数字格式化为具有指定位数精度的字符串：

12345.6789.toPrecision(4); // "1.235e+4"

**toString(radix)**

将数字转换为指定进制的字符串：

255.toString(16); // "ff"

**4. 其他方法**

**Number.isInteger()**

检查一个值是否为整数：

Number.isInteger(123); // true

Number.isInteger(123.0); // true

Number.isInteger(123.456); // false

**Number.parseInt(), Number.parseFloat()**

ES6 中提供的全局方法，与 parseInt 和 parseFloat 相似，但更加严格：

Number.parseInt('123.45', 10); // 123

Number.parseFloat('123.45abc'); // 123.45

**实例**

假设你需要将一个数字格式化为带有货币符号的字符串：

function formatCurrency(value) {

return '$' + value.toFixed(2).replace(/\B(?=(\d{3})+(?!\d))/g, ',');

}

console.log(formatCurrency(1234567.89)); // 输出 "$1,234,567.89"

这些方法覆盖了大多数与数字相关的常见操作需求，从基本的数学运算到复杂的格式化和验证。在实际开发中，根据具体情况选择合适的方法可以提高代码的可读性和效率。