数值

> - JavaScript 对 15 位的十进制数都可以精确处理，大于 2 的 53 次方的数值，都无法保持精度。
> - JavaScript 提供 Number 对象的 `MAX_VALUE` 和 `MIN_VALUE` 属性，返回可以表示的具体的最大值和最小值。

1. 数值表示法

以下两种情况，JavaScript 会自动将数值转化为科学计数法，其他情况都采用字面形式直接表示。

（1）小数点后的 0 多于 5 个

```
0.0000003 // 3e-7
0.000003  // 0.000003
```

（2）小数点前的数字多于 21 位

```
123456789012345678901  // 1.2345678901234568e+21
12345678901234567901   // 123456789012345680000
```

2. 特殊数值

2.1. `NaN`

`NaN` 是 JavaScript 的特殊值，表示“非数字”（`Not a Number`），主要出现在将字符串解析成数字出错的场合。

```
typeof NaN; // 'number'
+'x';       // NaN
```

0 除以 0 也会得到 `NaN`。

```
0 / 0; // NaN
```

一些数学函数的运算结果会出现 `NaN`。

```
Math.sqrt(-1); // NaN
```

`NaN` 不等于任何值，包括它本身。

```
NaN === NaN; // false
```

NaN 在布尔运算时被当作 false。

```
Boolean(NaN);       // false
Boolean(0);         // false
Boolean('');        // false
Boolean(false);     // false
Boolean(undefined); // false
Boolean(null);      // false
Boolean({});        // true
Boolean([]);        // true
```

NaN 与任何数（包括它自己）的运算，得到的都是 NaN。

```
NaN + 32;  // NaN
NaN + NaN; // NaN
```

2.2. Infinity

Infinity 加上或乘以 Infinity，返回的还是 Infinity。Infinity 减去或除以 Infinity，得到 NaN。0 乘以 Infinity，返回 NaN；0 除以 Infinity，返回 0。

```
Infinity + Infinity; // Infinity
Infinity - Infinity; // NaN
Infinity * Infinity; // Infinity
Infinity / Infinity; // NaN
0 * Infinity;        // NaN
0 / Infinity;        // 0
```

Infinity 与 null 计算时，null 会转成 0，等同于与 0 的计算。

```
null * Infinity; // NaN
null / Infinity; // 0
Infinity / null; // Infinity
```

Infinity 与 undefined 计算，返回的都是 NaN。

```
undefined + Infinity; // NaN
undefined - Infinity; // NaN
undefined * Infinity; // NaN
```

```
undefined / Infinity; // NaN
Infinity / undefined; // NaN
```

3. 与数值相关的全局方法

3.1. parseInt()

字符串转为整数的时候，是一个个字符依次转换，如果遇到不能转为数字的字符，就不再进行下去，返回已经转好的部分。

```
parseInt('8a');    // 8
parseInt('12**');  // 12
parseInt('12.34'); // 12
parseInt('15e2');  // 15
parseInt('15px');  // 15
```

如果字符串的第一个字符不能转化为数字（后面跟着数字的正负号除外），返回 NaN。

```
parseInt('abc'); // NaN
parseInt('.3');  // NaN
parseInt('');    // NaN
parseInt('+');   // NaN
parseInt('+1');  // 1
```

对于那些会自动转为科学计数法的数字，parseInt() 会将科学计数法的表示方法视为字符串，因此导致一些奇怪的结果。

```
parseInt(1000000000000000000000.5); // 1
// 等同于
parseInt('1e+21');                  // 1

parseInt(0.0000008); // 8
// 等同于
parseInt('8e-7');    // 8
```

parseInt() 会将空字符串转为 NaN。

```
parseInt(''); // NaN
```

3.2. parseFloat()

parseFloat() 会将空字符串转为 NaN。

```
parseFloat(''); // NaN
```

`parseFloat()` 的转换结果不同于 `Number()` 函数。

```
parseFloat(true);      // NaN
Number(true);          // 1

parseFloat(null);      // NaN
Number(null);          // 0

parseFloat('');        // NaN
Number('');            // 0

parseFloat('123.45#'); // 123.45
Number('123.45#');     // NaN
```

3.3. `isNaN()`

`isNaN()` 只对数值有效，如果传入其他值，会被先转成数值。传入字符串的时候，字符串会被先转成 `NaN`，所以最后返回 `true`。也就是说，*`isNaN()` 为 `true` 的值，有可能不是 `NaN`，而是一个字符串。*

```
isNaN('Hello');         // true
// 相当于
isNaN(Number('Hello')); // true
```

使用 `isNaN()` 之前，最好判断一下数据类型。`NaN` 唯一不等于自身这个特点。

```
function myIsNaN(value) {
  return value !== value;
}
```

3.4. `isFinite()`

`isFinite()` 方法返回一个布尔值，表示某个值是否为正常的数值。

```
isFinite(Infinity);  // false
isFinite(-Infinity); // false
isFinite(NaN);       // false
isFinite(undefined); // false
isFinite(null);      // true
isFinite(-1);        // true
```