Числа

В современном JavaScript существует два типа чисел:

- Обычные числа в JavaScript хранятся в 64-битном формате IEEE-754, который также называют «числа с плавающей точкой двойной точности» (double precision floating point numbers). Для хранения числа используется 64 бита: 52 из них используется для хранения цифр, 11 из них для хранения положения десятичной точки (если число целое, то хранится 0), и один бит отведён на хранение знака.
- BigInt числа дают возможность работать с целыми числами произвольной длины. Они нужны достаточно редко и используются в случаях, когда необходимо работать со значениями более чем 2^{53} или менее чем $-2^{53.}$

```
const bigint = 1234567890123456789012345678901234567890n;
const sameBigint = BigInt("1234567890123456789012345678901234567890");
const bigintFromNumber = BigInt(10); // то же самое, что и 10n
```

Примеры чисел:

```
alert( 7.3e9 );  // 7.3 миллиардов (7,300,000,000)
let ms = 1e-6; // шесть нулей, слева от 1
alert( 0xff ); // 255 шестнадцатиричная форма
let a = 0b11111111; // бинарная форма записи числа 255
let b = 0o377; // восьмеричная форма записи числа 255
```

Специальные числовые значения:

- Infinity (и -Infinity) — особенное численное значение, которое ведёт себя в точности как математическая бесконечность ∞.
- NaN представляет ошибку.

num.toString(base) возвращает строковое представление числа num в системе счисления base (base от 2 зо 36, по умолчанию 10).Например:

```
let num = 255;
alert( num.toString(16) );  // ff
alert( num.toString(2) );   // 11111111
alert( 123456..toString(36) ); // 2n9c
```

Внимание! Две точки в 123456..toString(36) это не опечатка. Если нам надо вызвать метод непосредственно на числе, как toString в примере выше, то нам надо поставить две точки .. после числа.
Также можно записать как
(123456).toString(36).

Math.floor

Округление в меньшую сторону: 3.1 становится 3, а -1.1 — -2.

Math.ceil

Округление в большую сторону: 3.1 становится 4, а -1.1 — -1.

Math.round

Округление до ближайшего целого: 3.1 становится 3, 3.6 — 4, а -1.1 — -1.

Math.trunc (не поддерживается в Internet Explorer)

Производит удаление дробной части без округления: 3.1 становится 3, а -1.1 — -1.

	Math.floor	Math.ceil	Math.round	Math.trunc
3.1	3	4	3	3
3.6	3	4	4	3
-1.1	-2	-1	-1	-1
-1.6	-2	-1	-2	-1

Метод toFixed(n) округляет число до n знаков после запятой и возвращает строковое представление результата.

```
let num = 12.34;
alert( num.toFixed(1) ); // "12.3"
```

Округляет значение до ближайшего числа, как в большую, так и в меньшую сторону, аналогично методу Math.round:

Наиболее часто встречающаяся ошибка при работе с числами в JavaScript – это потеря точности.
```
alert( 0.1 + 0.2 == 0.3 ); // false 0.30000000000000004
```

Решение проблемы точности:

Одно из решений - прибавить к исходному числу Number.EPSILON.

isNaN(value) преобразует значение в число и проверяет является ли оно NaN:
```
alert( isNaN(NaN) ); // true
alert( isNaN("str") ); // true
```
Нужна ли нам эта функция? Разве не можем ли мы просто сравнить === NaN? К сожалению, нет. NaN является неравным (посредством сравнения через ==, !=, ===, and !==) любому другому значению, включая другое значение NaN.

isFinite(value) преобразует аргумент в число и возвращает true, если оно является обычным числом, т.е. не NaN/Infinity/-Infinityю.
Иногда isFinite используется для проверки, содержится ли в строке число:
```
let num = +prompt("Enter a number", '');
// вернёт true всегда, кроме ситуаций, когда аргумент - Infinity/-Infinity или не число
alert( isFinite(num) );
```

Помните, что пустая строка интерпретируется как 0 во всех числовых функциях, включая isFinite.

Для явного преобразования к числу можно использовать + или Number(). Если строка не является в точности числом, то результат будет NaN:
```
alert( +"100px" ); // NaN
```

В реальной жизни мы часто сталкиваемся со значениями у которых есть единица измерения, например "100px" или "12pt" в CSS. Также во множестве стран символ валюты записывается после номинала "19€". Так как нам получить числовое значение из таких строк?

Для этого есть parseInt и parseFloat.
Функция parseInt возвращает целое число, а parseFloat возвращает число с плавающей точкой. Функция parseInt() имеет необязательный второй параметр. Он определяет систему счисления
```
alert( parseInt('100px') ); // 100
alert( parseFloat('12.5em') ); // 12.5
alert( parseInt('12.3') ); // 12, вернётся только целая часть
alert( parseFloat('12.3.4') ); // 12.3, произойдёт остановка чтения на второй точке
alert( parseInt('a123') ); // NaN, на первом символе происходит остановка чтения
```

Разбить число на разряды (свойство useGrouping):

1 вариант:
```
console.log((125452.32).toLocaleString("ru-RU",{useGrouping:true})); //"125 452,32"
```

2 вариант:
```
const priceFormatter = new Intl.NumberFormat();
priceFormatter.format(нужное число);
```

Math.random()

Возвращает псевдослучайное число в диапазоне от 0 (включительно) до 1 (но не включая 1)
```
alert( Math.random() ); // 0.1234567894322
```

Math.max(a, b, c...) / Math.min(a, b, c...)

Возвращает наибольшее/наименьшее число из перечисленных аргументов.
```
alert( Math.max(3, 5, -10, 0, 1) ); // 5
```

```
alert( Math.min(1, 2) ); // 1
```

Math.pow(n, power)

Возвращает число n, возведённое в степень power

```
alert( Math.pow(2, 10) ); // 2 в степени 10 = 1024
```