Number (числа)

В JavaScript в отличие от некоторых других языков имеется только один числовой тип **`number`**.
В нём используется формат чисел с плавающей точкой двойной точности по стандарту IEEE 754.
То есть, в JavaScript нет отдельного формата для целых чисел. Все числа в JavaScript имеют плавающую точку.

Пример

```
let a = 50; // целое число
let b = 4.7; // число с дробной частью
let c = 2e3; // в экспоненциальной форме 2*10^3 (2000)
let d = 5.8e-2; // в экспоненциальной форме 5.8*10^-2 (0,058)
let e = 010; // в восьмеричной системы счисления
let f = 0xFF; // в шестнадцатеричной системе счисления
```

Специальные значения Number

Кроме чисел, формат `number` также содержит специальные числовые значения:

- **«NaN»** (Not-a-Number) применяется при невозможности преобразовать результат к числу, то есть результатом не мб число, по сути, обозначает ошибку; console.log(0/0),console.log(3 - '1rem');
- **«Infinity»** — для представления положительной бесконечности (+∞) ;console.log(10/0)
- «-**Infinity»** — для представления отрицательной бесконечности (–∞) console.log(-10/0);

Округление

Math.floor	Округление в меньшую сторону	`Math.floor(3.1);`	3
Math.ceil	Округление в большую сторону	`Math.ceil(3.1);`	4
Math.round	Округление до ближайшего целого, в большую сторону, если дробная часть `>=` 0.5; иначе в меньшую сторону;	`Math.round(3.1);`	3

Точное округления метод toFixed(n)

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

```
let num = 12.34;

num.toFixed(1); // вернет строку "12.3"

+num.toFixed(1); // 12.3 строка
```

Задание

1) Запросите у пользователя две стороны прямоугольника и выведите периметр, площадь фигуры

   P=2(A+B); S=AB

2) Запросите у пользователя расстояние между городами (км) и время (мин), за которое необходимо преодолеть расстояние. Выведите сообщение, с какой скоростью необходимо перемещаться(км/ч)

   S=VT

```
let = +prompt("Введите длину стороны квадрата","");
              alert("Периметр квадрата равен " + l*4);


let  s = +prompt("Введите расстояние между городами, которое Вы хотите проехать","");
              let  t = +prompt("Введите время, за которое Вы хотите преодолеть указанное расстояние (ч)","");
               let  v = s/(t/60);
               alert("Вам необходимо ехать со скоростью хотя бы " + v + "км/ч, чтоб успеть вовремя!");
```

Операторы Javascript

В JavaScript существует множество различных операторов:

- арифметические операторы,
- операторы присваивания,
- строковые операторы,
- операторы сравнения,
- логические операторы,
- операторы типов,
- побитовые операторы.

Оператор

Оператор – это символ(ы) или ключевое слово, благодаря которому производятся некоторые виды вычислений с участием одного или нескольких значений. Значения, располагающиеся слева и справа от оператора, называются **операндами**. Иногда операнды называют ещё **аргументами**.

Оператор присваивания (=) присваивает значение переменной

оператор присваивания используется с двумя операндами

Арифметические операторы JavaScript

Арифметические операторы используются для выполнения арифметических операций с числами:

Оператор	Описание	Примеры
+	Сложение	x+y
-	Вычитание	x-y
*	Умножение	x*y
/	Деление	x/y
%	Остаток от деления	x%y
**	Возведение в степень	x**y
++	Инкремент(Увеличение на 1)	x+1
--	Декремент(Уменьшение на 1)	x-1

Таблица приоритетов операций в порядке их убывания

название	обозначение
инкремент	++
декремент	--
отрицание	!
унарный минус	-
умножение	*
деление, остаток от деления	/,%
сложение	+
вычитание	-
сравнение	<, >, <=, >=
равенство	==
не равенство	!=
логическое И	&&
логическое ИЛИ	||
присваивание	=, +=, -=, *=, /=, %=, !=

Приоритеты

Все операции выполняются слева направо согласно приоритету. Те операции, у которых приоритет выше, выполняются раньше

```
x = 1 + 2 * z - 5

x = y - (z+100 / 4) * 2


a = b = 5; //Сначала b становится равным 5, затем a принимает значение b.

let a=1, b=2, c=3;

d=a=b=c;  // d = ?
```

Список сокращений операторов присваивания

x = y	x = y	Присваивание
x += y	x = x + y	Присваивание со сложением
x -= y	x = x - y	Присваивание с вычитанием
x *= y	x = x * y	Присваивание с умножением
x /= y	x = x / y	Присваивание с делением

```
let n = 2;

n = n + 5;

n = n * 2;
```

Эту запись можно укоротить при помощи совмещённых операторов += и *=:

```
let n = 2;

n += 5; // теперь n = 7 (работает как n = n + 5)

n *= 2; // теперь n = 14 (работает как n = n * 2)
```

Методы объекта Math

Метод	Описание
Math.cos(x)	Возвращает косинус угла x (x должен быть в радианах)
Math.exp(x)	Возвращает экспоненту от x (Ex)
Math.log(x)	Возвращает натуральный логарифм (по основанию E) числа x
Math.max(x, y, z, ..., n)	Возвращает наибольшее значение в списке
Math.min(x, y, z, ..., n)	Возвращает наименьшее значение в списке
Math.sin(x)	Возвращает синус угла x (x должен быть в радианах)
Math.PI	Сокращенная запись популярного числа Пи 3.14159
Math.E	число Эйлера

Методы объекта Math

Метод	Описание
Math.max(x, y, z, ..., n)	Возвращает наибольшее значение в списке
Math.min(x, y, z, ..., n)	Возвращает наименьшее значение в списке
Math.random()	Возвращает случайное число между 0 и 1
Math.round(x)	Возвращает округленное значение x
Math.sqrt()	возвращает квадратный корень из x
Math.pow(x, y)	Возводит значение x в степень y
Math.abs(x)	Возвращает абсолютное значение от x

Если метод Math.random() использовать вместе с методом Math.floor(),
то можно генерировать случайные целые числа.

```
Math.floor(Math.random() * 10);       // возвращает число от 0 до 9

Math.floor(Math.random() * 11);       // возвращает число от 0 до 10

Math.floor(Math.random() * 100);      // возвращает число от 0 до 99

Math.floor(Math.random() * 101);      // возвращает число от 0 до 100

Math.floor(Math.random() * 10) + 1;   // возвращает число от 1 до 10

Math.floor(Math.random() * 100) + 1; // возвращает число от 1 до 100
```

Задание

1) Запросите у пользователя радиус окружности и выведите длину окружности, площадь окружности

$L=2\pi R$; $S = \pi R$

```
let  r = +prompt("Введите радиус окружности","");

               let  s = Math.PI * Math.pow(r,2);

                alert("Площадь окружности равна " + s.toFixed(3));
```

Задание

2) необходимо составить программу для расчета формулы:

$$f = x + \frac{2}{x+1} \cdot y \,.$$

Задание

1) Ввести три числа и вывести наименьшее (используя Math)
2) Вывести наименьшее случайное число из трех случайных чисел в диапазоне от 1 до 10 (`Math.floor(Math.random() * 10) + 1;`)
3) Зарплата работника составляет $250 + 10% от продаж. Запросите общую сумму продаж за месяц и посчитайте зарплату.
4) Запросить трехзначное число и вывести последнюю цифру, вторую цифру, первую цифру (`использовать оператор - остаток от деления %`)
5) Запросить трехзначное число и вывести задом наперед
6) Запросите у пользователя пятизначное число и переместите последнюю цифру в начало (`из числа 12345 должно получиться число 51234`).

Оператор

- с одним операндом называется **унарным**,
- с двумя – **бинарным**,
- с тремя – **тернарным (** условный оператор.**)**.

Бинарная операция использует два операнда, один перед оператором и другой за ним:

operand1 operator operand2

Например:

```
3+4
x*y
d = 'Привет'
f1 && f2
```

унарная операция использует один операнд, перед или после оператора:

operator operand	*operand operator*

Например:

x++

++x

x=-x

Чему будут равны переменные `a` и `x` после исполнения кода в примере ниже?

```
let a = 2;

let x = 1 + (a *= 2);
```

Строки (`string`)

В JavaScript строка представляет собой последовательность из одного или нескольких символов, обычно состоящую из символов, букв и цифр. Строка в JavaScript должна быть заключена в кавычки.

В JavaScript существует три типа кавычек.

1. Двойные кавычки: "Привет".
2. Одинарные кавычки: 'Привет'.
3. Обратные кавычки: `Привет`.

Двойные или одинарные кавычки являются «простыми», между ними нет разницы в JavaScript.

Строка: Интерполяция

Интерполяция строк — это процесс, используемый для оценки строкового литерала, который содержит один или несколько заполнителей.

Таким образом, заполнители заменяются соответствующими значениями, что делает модель форматирования строк более интуитивно понятной и простой. Имея это в виду, давайте выясним, как работает интерполяция строк JavaScript и какие стандарты используются. Интерполяция строк обозначается обратными кавычками, например ``.

Обратные кавычки позволяют нам встраивать выражения в строку, заключая их в `${…}`

```
let cucumbers = 5, tomatoes = 7;

console.log(`У меня есть ${cucumbers} огурцов`);

console.log(`У меня есть ${cucumbers + tomatoes} овощей`);
```

Выражение внутри `${…}` вычисляется, и его результат становится частью строки.

Какой результат будет у выражений ниже?

```
6 / "3"

"2" * "3"

4 + 5 + "px"

"$" + 4 + 5

"4" - 2

"4px" - 2

"-9" + 5

"9" - 5

16 + 4 + "Volvo"

"Volvo" + 16 + 4;
```

! При сложении числа и строки JavaScript будет воспринимать число как строку.
JavaScript вычисляет выражения слева направо
Таким образом, разные последовательности могут привести к разным результатам:

Что выведет этот скрипт?

```
let name = "Ilya";


alert( `hello ${1}` ); // ?

alert( `hello ${"name"}` ); // ?

alert( `hello ${name}` ); // ?
```

Приведение типов в JavaScript

toNumber

В Javascript преобразование чисел, строк, объектов к числу (не обязательно целому, может быть и дробное) можно сделать с помощью функции `Number()`:

```
Number(myVar);
```

Входной тип	Результат
Undefined	NaN
Null	0
Boolean	Результат равен 1, если аргумент равен true. Результат равен 0, если аргумент равен false.
Number	Результат совпадает с входным аргументом (преобразование не производится).
String	1. Для строкового численного литерала вернёт исходное число. 2. Для не численного литерала вернёт NaN.
Object	Применяются следующие шаги: 1. Вызвать ToPrimitive (входной аргумент, подсказка Number). 2. Вызвать ToNumber(Результат(1)). 3. Вернуть Результат(2).

```
Number(5); //5
Number('0.25'); //0.25
Number('q5'); //NaN
Number('abc'); //NaN
Number(false); //0
Number(true); //1
```

Преобразование к целочисленному типу (parseInt)

parseInt(string, radix);

string - строковое представление числа

radix - основание системы счисления

Функция parseInt преобразует первый аргумент в число по указанному основанию, а если это невозможно - возвращает NaN.

Например, radix=10 даст десятичное число, 16 - шестнадцатиричное и т.п. Для radix>10 цифры после девяти представлены буквами латинского алфавита.

Если в процессе преобразования parseInt обнаруживает цифру, которая не является цифрой в системе счисления с основанием radix, например G вв 16-ричной системе или A в десятичной, то процесс преобразования тут же завершается и возвращается значение, полученное из строки на данный момент.

parseInt округляет дробные числа, т.к останавливается на десятичной точке.

Если radix не указан или равен 0, то javascript предполагает следующее:

- Если входная строка начинается с "0x", то radix = 16
- Если входная строка начинается с "0", то radix = 8. Этот пункт зависит от реализации и в некоторых браузерах (Google Chrome) отсутствует.
- В любом другом случае **radix=10**

Если преобразовать в число не удается, parseInt возвращает NaN

Важный факт, что функция позволяет использовать пробелы в начале и конце входной строки. Кроме того `parseInt()` округляет дробные числа, потому что в работе останавливается на дробной десятичной точке.

```
parseInt(5); //5
parseInt(5.33); //5
parseInt(' 5.64'); //5
parseInt(true); //NaN
parseInt(false); //NaN
parseInt('abc'); //NaN
parseInt('5abc'); //5
parseInt('x5abc'); //NaN
parseInt('123bc',10) // 123
```

Булевый (логический) тип

Булевый тип (`boolean`) может принимать только два значения: `true` (истина) и `false` (ложь).

Такой тип, как правило, используется для хранения значений да/нет: `true` значит «да, правильно», а `false` значит «нет, не правильно».

```
let nameFieldChecked = true; // да, поле отмечено

let ageFieldChecked = false; // нет, поле не отмечено
```

Булевые значения также могут быть результатом сравнений:

```
let isGreater = 4 > 1;


alert( isGreater ); // true (результатом сравнения будет "да")
```

toBoolean

В Javascript преобразование типа к boolean можно сделать следующими способами:

```
!!myVar
Boolean(myVar)
```

Оба предложенных варианта **toBoolean** ("к булевскому") преобразуют свой аргумент в значение типа Boolean согласно следующей таблице:

Входной тип	**Результат**
Undefined	false
Null	false
Boolean	Результат совпадает с входным аргументом (преобразование не производится).
Number	Результат равен false, если аргумент равен 0 или NaN, иначе результат равен true.
String	Результат равен false, если аргумент является пустой строкой (его длина равна нулю), иначе результат равен true.
Object	true

Использование методов `prompt` и `alert` для создания игры «Угадай число»?

```
const number = 7;
let result = false;

while (!result) {
  const answer = prompt('Угадай число от 1 до 10?');
  if (answer === null) {
    break;
  }
  switch (+answer) {
    case number - 2:
    case number + 2:
      alert('Уже теплее!');
      break;
    case number - 1:
    case number + 1:
      alert('Горячо!');
      break;
    case number:
      alert('Ты угадал! Это число {$number}.');
      result = true;
      break;
    default:
      alert('Холодно!');
  }
  }
```