JavaScript

Урок №5: Числа

Автор Бахшиллоев Бехзод
Ментор по Frontend в PRO UNITY

Числа

В современном JavaScript существует два типа чисел:

1. 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). Это числа, которые мы будем использовать чаще всего. Мы поговорим о них в этой главе.
2. 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 $(2^{53}-1)$ или менее чем $-(2^{53}-1)$. Так как BigInt числа нужны достаточно редко, мы рассмотрим их в отдельной главе BigInt.

Способы записи числа

Представьте, что нам надо записать число 1 миллиард. Самый очевидный путь:

```
let billion = 1000000000;
```

Мы также можем использовать символ нижнего подчёркивания _ в качестве разделителя:

```
let billion = 1_000_000_000;
```

Символ нижнего подчёркивания _ – это «синтаксический сахар», он делает число более читабельным. Движок JavaScript попросту игнорирует _ между цифрами, поэтому в примере выше получается точно такой же миллиард, как и в первом случае.

Однако в реальной жизни мы в основном стараемся не писать длинные последовательности нулей, так как можно легко ошибиться. Укороченная запись может выглядеть как "1млрд" или "7.3млрд" для 7 миллиардов 300 миллионов. Такой принцип работает для всех больших чисел.

В JavaScript, чтобы укоротить запись числа, мы можем добавить к нему букву "e" и указать необходимое количество нулей:

Способы записи числа

```
let billion = 1e9;  // 1 миллиард, буквально: 1 и 9 нулей

alert( 7.3e9 );  // 7.3 миллиарда (7,300,000,000)
```

Другими словами, "e" умножает число на 1 с указанным количеством нулей.

```
1e3 === 1 * 1000 // e3 означает *1000

1.23e6 === 1.23 * 1000000 // e6 означает *1000000
```

А сейчас давайте запишем что-нибудь очень маленькое. К примеру, 1 микросекунду (одна миллионная секунды):

```
let mcs = 0.000001;
```

В этом случае нам также поможет "e". Если мы хотим избежать записи длинной последовательности из нулей, мы можем сделать так:

```
let ms = 1e-6; // шесть нулей слева от 1
```

Если мы подсчитаем количество нулей в 0.000001, их будет 6. Естественно, верная запись 1e-6.

Числа

- let x = 10;
- let y = x + 10;
- let z = 3.14;
- parseInt(“200.5px”) // 200;
- parseFloat(“20.5%”) // 20.5;
- parseFloat(“is20.5%”) // NaN

Infinity - 1/0, 1e+309

NaN - 1 - ‘string’, 2*’10px’, 0/0

isNaN() - isNaN(‘string’) // true

isFinite() - isFinite(123) // true

isFinite(‘string’) // false

Полезные ссылки: Числа, NaN

Числа. Округление и объект Math.

Math - встроенный инструмент для работы с числам

```
Math.round(5.1) // 5
Math.round(5.5) // 6
Math.floor(20.1) // 20
Math.floor(20.5) // 20
Math.floor(20.99) // 20
Math.ceil(20.1) // 21
Math.ceil(20.5) // 21

(100.123).toFixed(2) // ‘100.12’
```

Задачи:

1. Получить число pi из Math и округлить его до 2-х знаков после точки
2. Используя Math, найти максимальное и минимальное числа из представленного ряда *15, 11, 16, 12, 51, 12, 13, 51*

3. Работа с Math.random:
    a. Получить случайное число и округлить его до двух цифр после запятой
    b. Получить случайное **целое число** от 0 до X. X - любое произвольное число.

4. Проверить результат вычисления 0.6 + 0.7 - как привести к нормальному виду (1.3)?
5. Получить число из строки ‘100$’