Frontender[1.0] JavaScript - Рекурсия, когда функция вызывает сама себя

YouTube	https://youtu.be/cwIBFzBcXKI
Telegram	https://t.me/Dmitry_Kolotilshikov
Github	https://github.com/DmitryKolotilshikov/
Boosty	https://boosty.to/dmitry_ko
# Номер урока	39

💡 **Задачи к этому уроку тут https://boosty.to/dmitry_ko**

💡 Полезные ссылки:

https://learn.javascript.ru/recursion

💡 В процессе выполнения задачи в теле функции могут быть вызваны другие функции для выполнения подзадач. Частный случай подвызова – когда функция вызывает ***сама себя***. Это как раз и называется ***рекурсией***.

Любая рекурсия может быть переделана в цикл. Как правило, вариант с циклом будет эффективнее.

Но переделка рекурсии в цикл может быть **нетривиальной**, особенно когда в функции в зависимости от условий используются различные рекурсивные подвызовы, результаты которых объединяются, или когда ветвление более сложное. Оптимизация может быть ненужной и совершенно нестоящей усилий.

Часто код с использованием **рекурсии более короткий**, **лёгкий** для понимания и поддержки. Оптимизация требуется не везде, как правило, нам важен хороший код, поэтому она и используется.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js"></script>
    <title>Recursion</title>
</head>
```

```
<body>
</body>
</html>
```

```
// Рекурсия (recursion)3

const log = console.log;

// 💎----------Возведение в натарульную степень----------💎

const pow1 = (x, n) => {
    let result = 1;

    for (let i = 0; i < n; i++) {
        result *= x;
    }

    return result;
}

log(pow1(2, 3));

// возведение в степень с рекурсией
const pow = (x, n) => {
    if (n === 1) {
        return x;
    } else {
        return x * pow(x, n - 1);
    }
}

log(pow(2, 3));

// 💎-------------Сумма чисел--------------💎

const sum1 = (n) => {
    let sum = 0;
    for (let i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}

log(sum1(5));


// подсчет суммы рекурсивно
const sum = (n) => {
    if (n === 1) return 1;

    return n + sumTo(n - 1);
}

log(sum(5)); // 1 + 2 + 3 + 4 + 5
log(sum(6)); // 1 + 2 + 3 + 4 + 5 + 6
log(sum(100));
```

```
// подсчет суммы математично, самый быстрый способ 😊
const sum3 = (n) => {
    return n * (n + 1) / 2;
}
log(sum3(100));

// 💎-------------Факториал--------------💎

/*
Факториал — это произведение всех натуральных чисел от 1
до данного числа. Например, факториал числа 5 будет равен
1 × 2 × 3 × 4 × 5 = 120
*/

const factorial = (n) => {
    return (n != 1) ? n * factorial(n - 1) : 1;
}

log(factorial(5)) // 120
```