**Рекурсия**

Рекурсия – это вызов функции внутри самой себя. Одну и ту же задачу можно реализовать циклом и рекурсией.

Общее количество вложенных вызовов (включая первый) называют глубиной рекурсии. В нашем случае она будет равна n. Максимальная глубина рекурсии ограничена движком JavaScript. Точно можно рассчитывать на 10000 вложенных вызовов. Глубина рекурсии равна максимальному числу контекстов, одновременно хранимых в стеке.

Контекст выполнения и стек

//цикл

*function* f2() {

*let* out ='';

    for(*let* i = 1; i <= 30; i++) {

        out += i + ' ';

    }

    console.log(out);

}

f2();

//рекурсия

*let* i = 0;

*let* out = '';

*function* f3() {

    i++;

    out += i + ' ';

    if(i >= 30) return;

    f3();

}

f3();

console.log(out);

Задача по получению прибыли

//Программа получения прибыли РЕКУРСИЯ

//Функция генерирующая случайное число в диапазоне min и max

*function* randomInteger(*min*, *max*) {

*let* rand = *min* + Math.random() \* (*max* + 1 - *min*);

    return Math.floor(rand);

}

*let* s1 = 0;

*function* moneyRecursion() {

    //Получаем деньги от 0 до 100

*let* money = randomInteger(0, 100);

    console.log('add' + money);

    //накапливаем

    s1 += money;

    console.log('sum: ' + s1);

    //Если сумма больше, то останавливаемся

    if(s1 >= 300) return;

    moneyRecursion();

}

moneyRecursion();

//ЦИКЛ

*function* randomInteger(*min*, *max*) {

*let* rand = *min* + Math.random() \* (*max* + 1 - *min*);

    return Math.floor(rand);

}

*function* moneyCycle() {

*let* s2 = 0;

    for(*let* i = 0; true; i++) {

*let* money = randomInteger(0, 100);

        console.log('add' + money);

        s2 += money;

        console.log('sum: ' + s2);

        if(s2 >= 300) return;

    }

}

moneyCycle();

В данном примере приведен бесконечный цикл с параметром true

Перебор объектов с помощью рекурсии

Каждый человек описан двумя критериями. Это возраст (age) и если есть родители (parent). Кол-во вложений никак не ограничено.

Пусть необходимо вывести фамилию всех родителей

*const* users = {

    "ivanov": {

        age: 25,

        parent: {

            "ivanov-a": {

                age: 45

            },

            "ivanov-b": {

                age: 43,

                parent: {

                    "sergeev-a": {

                        age: 88,

                        parent: {

                            "lionenko": {}

                        }

                    }

                }

            }

        }

    },

    "kostenko": {

        age: 56,

        parent: {

            "ignatenko": {

            },

            "sniezko": {

                age: 45

            }

        }

    }

}

*function* userParentRecursion(*obj*) {

    if(*obj*.parent !== undefined) {

        for(*let* key in *obj*.parent) {

            console.log(key);

            userParentRecursion(*obj*.parent[key]);

        }

    }

}

for (*let* key in users) {

    userParentRecursion(users[key]);

}

Результат



Пример

Необходимо подсчитать суммарную зарплату всех сотрудников

*let* company = { // тот же самый объект, сжатый для краткости

    sales: [{name: 'John', salary: 1000}, {name: 'Alice', salary: 600 }],

    development: {

      sites: [{name: 'Peter', salary: 2000}, {name: 'Alex', salary: 1800 }],

      internals: [{name: 'Jack', salary: 1300}]

    }

  };

  // Функция для подсчёта суммы зарплат

*function* sumSalaries(*department*) {

    console.log(*department*);

    debugger;

    if (*Array*.isArray(*department*)) { // случай (1)

      return *department*.reduce((*prev*, *current*) *=>* *prev* + *current*.salary, 0); // сумма элементов массива

    } else { // случай (2)

*let* sum = 0;

      for (*let* subdep of *Object*.values(*department*)) {

        sum += sumSalaries(subdep); // рекурсивно вызывается для подотделов, суммируя результаты

      }

      return sum;

    }

  }

  alert(sumSalaries(company)); // 6700

Пример

Функция возведения в степень при помощи цикла и рекурсии

*const* pow = (*x*, *n*) *=>* {

*let* result = 1;

    for(*let* i = 0; i < *n*; i++) {

        result \*= *x*;

    }

    return result;

};

console.log(pow(2, 4));

*const* pow = (*x*, *n*) *=>* {

    if(*n* == 1) {

        return *x*;

    } else {

        return *x* \* pow(*x*, *n* - 1); 2\* 2\* 2\* 2 // 4/8/16

    }

}

console.log(pow(2, 4));

