Лекция:

*Кекс задумчиво махал хвостом и пристально щурился в ваш код. Затем он оторвал усталый взгляд от монитора, набрал на пейджере знакомый номер и, вздохнув, сказал: «Тимлид, привет, расскажи новичку про рефакторинг».*

Удивительно, но видеоблог смог просуществовать достаточно долго, и сегодня Кекс прислал данные за последний месяц:

817, 1370, 752, 1247, 681, 1120, 915, 1281, 875, 1341, 757, 610, 812, 1170, 769, 1261, 845, 1289, 515, 1247, 845, 1311, 741, 1239, 812, 638, 877, 1242, 1159, 1372

Чтобы проанализировать эти данные, придётся добавить в текущую программу ещё 26 переменных. Много писанины? А скоро данные и за год подоспеют.

Ваша программа работает правильно, но дополнять её новыми данными крайне трудоёмко. Поэтому пришло время рефакторинга. [Рефакторинг](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) — это переписывание программы, после которого она должна работать так же, но быть более гибкой. Цель нашего рефакторинга — снизить трудоёмкость добавления и изменения массивного количества данных.

Благо, существует способ хранить *массивные* данные. Этот способ так и называется — массив. Массив — это [тип данных](https://htmlacademy.ru/courses/207/run/4), который представляет собой список элементов, у каждого из которых есть свой порядковый номер.

Массивы создаются с помощью так называемого *литерала массива* — квадратных скобок. Внутри скобок через запятую перечисляются все значения, которые должен содержать массив. При создании массив, как любое значение, можно записать в переменную:

let numbers = [1, 2, 3, 4, 5];

Вообще, в массиве можно хранить любые данные: строки, булевы значения, числа и даже другие массивы.

Рефакторинг мы будем делать плавно, не удаляя всю программу, а переписывая кусками. На первом этапе избавимся от отдельных переменных для данных:

1. Временно присвоим среднему значению нулевое значение.
2. Создадим массив, в который перенесём значения из переменных.
3. Этот массив передадим в команду keks.plot вместо переменных (на счастье, команда умеет работать и с массивами).
4. И теперь, когда переменные для данных в программе больше не используются, смело их удаляем.

Задание

1. Вместо расчёта среднего значения присвойте переменной averageUsers значение 0
2. Перед командой console.log, где рисуем график добавьте переменную usersByDay со значением [812, 1360, 657, 1247]
3. Внутри console.log, где рисуем график, первым параметром задайте usersByDay, вторым expectedUsers, остальные параметры удалите.
4. Смело удаляйте переменные firstDayUsers, secondDayUsers, thirdDayUsers и fourthDayUsers

Код:

Script.js

let expectedUsers = 1000;

let firstDayUsers = 812;

let secondDayUsers = 1360;

let thirdDayUsers = 657;

let fourthDayUsers = 1247;

// Рисуем график посещаемости

console.log(firstDayUsers, secondDayUsers, thirdDayUsers, fourthDayUsers, expectedUsers);

var averageUsers = (firstDayUsers + secondDayUsers + thirdDayUsers + fourthDayUsers) / 4

console.log('Средняя посещаемость: ' + averageUsers);

if (averageUsers > expectedUsers) {

console.log('Посещаемость великолепна. Продолжай в том же духе!');

} else {

console.log('Посещаемость так себе. Нужно поднапрячься!');

}

Вывод:

812 1360 657 1247 1000

Средняя посещаемость: 0

Посещаемость так себе. Нужно поднапрячься!

**Решение**

let expectedUsers = 1000;

let usersByDay = [812, 1360, 657, 1247];

// Рисуем график посещаемости

console.log(

usersByDay,

expectedUsers

);

// Рассчитываем среднее значение посещаемости

let averageUsers = 0;

console.log('Средняя посещаемость: ' + averageUsers);

if (averageUsers > expectedUsers) {

console.log('Посещаемость великолепна. Продолжай в том же духе!');

} else {

console.log('Посещаемость так себе. Нужно поднапрячься!');

}