Frontender[1.0] JavaScript - Циклы. FOR, WHILE, база, бесконечный цикл, пузырьковая сортировка (bubble sort)

YouTube	https://youtu.be/cTaqQG5CNh8
Telegram	https://t.me/Dmitry_Kolotilshikov
Github	https://github.com/DmitryKolotilshikov/
Boosty	https://boosty.to/dmitry_ko
# Номер урока	21

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

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

```
// Циклы, for, while
/*
Для многократного повторения одного участка кода предусмотрены циклы.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10
и для каждого выполнить одинаковый код.
*/

const styles = `background: lightsalmon;`
const log = console.log;

// --------- Цикл for; ---------
```

```
const cars = ["audi", "ford", "mercedes", "mazda", "tesla"];

// log("1 " + cars[0]);
// log("2 " + cars[1]);
// log("3 " + cars[2]);
// log("4 " + cars[3]);
// log("5 " + cars[4]);

/*
for (стартовое выражение (начало); условное выражение (условие); инкремент или декремент (на каждом шаге)) {
    // логика на каждом шаге (итерации)
}
*/

// for (let i = 0; i <= 5; i++) {
//     log("-->", i);
// }

// let i = 0;
// for (; i < 5; i++) {
//     log("-->", i);
// }

// let i = 0;
// for (; i < 5;) {
//     log("-->", i++);
// }

// for (;;) {
//     log("вечный цикл");
// }

// ------------------------------------------

// for (let i = 0; i < cars.length; i++) {
//     log(`${i + 1} ${cars[i]}`);
// }

// for (let i = cars.length - 1; i >= 0; i--) {
//     log(`${i + 1} ${cars[i]}`);
// }

// ------------- Continue & break ------------

// for (let i = 0; i < cars.length; i++) {
//     if (cars[i] === "mercedes") continue;
//     log(`${i + 1} ${cars[i]}`);
// }

// for (let i = 0; i < cars.length; i++) {
//     if (cars[i] === "mercedes") break;
//     log(`${i + 1} ${cars[i]}`);
// }

// ------------- Цикл с массивом объектов ------------
```

```
const users = [
    { id: 1, name: "Alex", age: 35, position: "manager" },
    { id: 2, name: "Kate", age: 22, position: "qa" },
    { id: 3, name: "Nikita", age: 29, position: "developer" },
];

for (let x = 0; x < users.length; x++) {
    console.group(`Employee ${x + 1}`);
    log(`Name: ${users[x].name}`);
    log(`Age: ${users[x].age}`);
    log(`Position: ${users[x].position}`);
    console.groupEnd();
}

// -------------- Задачи ------------------

// 1) вычислить сумму всех чисел
const numbers1 = [1, 2, 3, 4, 5];

let sum = 0;

for (let i = 0; i < numbers1.length; i++) {
  sum += numbers1[i];
}

// 2) посчитать кол-во четных чисел в массиве

const numbers2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

let evenCount = 0;

for (let i = 0; i < numbers2.length; i++) {
  if (numbers2[i] % 2 === 0) {
    evenCount++;
  }
}

// ------------- Вложенный цикл -----------

// for (let i = 0; i < 3; i++) {
//     log(`%c${i + 1} - loop[1]`, styles);

//     for (let j = 0; j < 3; j++) {
//         log(`${j + 1} - loop[2]`);
//     }
// }

// -------------- Цикл While -----------

// let i = 0;
// while (i < cars.length) {
//     log(`while - ${i + 1}`);
//     i++;
// }

/*
while(true) используется
 - в алгоритмах где с неопределенным кол-вом итераций
```

```
 - в серверных приложениях, где например обрабатываются события или сообщения, когда
   неизвестно когда поступит следующее событие и нужно постоянно проверять новые данных
 - постоянный опрос какого-либо состояния, например сетевого соединения или устройства
   до тех пор пока не будет выполнено определенное условие для выхода из цикла
 - есть и другие случаи, всегда зависит от задачи
*/

// let i = 0;
// while (true) {
//     log("while(true)");
//     if (i >= 10) {
//         log(`i = ${i}`);
//         break;
//     }
//     i++;
// }

// checkConnection();

// -----------------do while---------------------

/*
do while -> тот, же while, только тело цикла идет не внутри while, а внутри do
Сегодня обычно используется обыный while, без do
*/

// let i = 0;
// do {
//     log(`do-while - ${i + 1}`);
//     i++;
// } while (i < cars.length);

// -------- Пузырьковая сортировка массива (bubble sort) ----
/*
    Простой и наглядный алгоритм для понимая.
    1) Проходим по массиву и сравниваем каждую пару соседних элементов.
    2) Если текущий элемент больше следующего, меняем их местами.
    3) Повторям процесс для всего массива, уменьшая кол-во проверяемых элементов
       на одну позицию с каждым шагом (итерацией), так как самый большой элемент
       всплывает в конец массива
*/

const array = [4, 33, 2, 1];

for (let i = 0; i < array.length; i++) {
    log(`%c${i + 1} - loop[1]`, styles);

    for (let j = 0; i < array.length - 1 - i; j++) {
        log(`${j + 1} - loop[2] ${array[j]} > ${array[j + 1]}`);

        if (array[j] > array[j + 1]) { // > по возрастанию; < по убыванию;
            const temp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = temp;
        }
    }
}
```

```
const array2 = [4, 33, 2, 1];

log(array);
log(array2.sort((a, b) => a - b));
log(array2.sort((a, b) => b - a));
log(array2.sort((a, b) => a - b).reverse());
```

```
// while(true) mock function example
let count = 0;
/*
тут
- замыкания (переменная count)
- асинхронные функции (pollNetwork и checkConnection)
- Promise и setTimeout для фейковой задержки
*/
const pollNetwork = async (delay = 1000) => {
    await new Promise(resolve => {
        setTimeout(resolve, delay);
    })

    count++;

    if (count < 5) {
        return { status: "pending" };
    }

    return { status: "connected" };
}

const checkConnection = async () => {
    while (true) {
        const response = await pollNetwork(1500);

        console.log(`STATUS = ${response.status}`);

        if (response.status === "connected") {
            break;
        }
    }
}

// checkConnection();
```