**Лабораторная работа №4**

**Тема работы:** Работа с циклами.

**Цель работы:** изучить особенности разных операторов циклов, их особенности и способы работы с ними.

**Теоретические сведения**

**Циклы** используются для того, чтобы некоторый участок кода выполнился несколько раз подряд.

**Зачем это нужно** - представьте, что вам нужно возвести в квадрат 100 элементов массива. Если обращаться к каждому элементу отдельно по его ключу - это займет 100 строчек кода, и для того, чтобы написать этого код, нужно будет потратить довольно много времени.

Но это не нужно - у нас есть возможность сделать так, чтобы JavaScript за нас выполнил некоторую операцию **нужное количество раз**. Например, возвел все элементы массива в квадрат.

Делается это с помощью **циклов**.

## Цикл while

Цикл **while** будет выполняться до тех пор, пока **верно** (истинно) выражение, переданное ему параметром. Смотрите синтаксис:

while ( пока выражение истинно ) {

выполняем этот код циклически;

в начале каждого цикла проверяем выражение в круглых скобках

}

/\*

Цикл закончится, когда выражение перестанет быть истинным.

Если оно было ложным изначально - то он не выполнится ни разу!

\*/

В принципе, цикл **while** может выполняться **бесконечно** (но это приведет к зависанию скрипта!), достаточно передать ему выражение, которое **никогда не станет ложным**. Например, так:

var test **=** true;

while (test **===** true) {

/\*

Написанный здесь код будет выполняться 'вечно'

(пока скрипт не будет остановлен принудительно).

Не стоит это повторять - это приведет к зависанию сервера!

\*/

}

Давайте последовательно выведем с помощью цикла while числа от одного до пяти:

var i **=** 0; //счетчик цикла

while (i **<** 5) {

/\*

С помощью оператора ++ увеличиваем i на единицу

при каждом проходе цикла.

\*/

i**++**;

alert(i);

}

Обратите внимание на переменную **i** – она является так называемым **счетчиком цикла**. Счетчики используются для того, чтобы подсчитывать, сколько раз выполнился цикл. Кроме того, они выполняют вспомогательную роль - в нашей задаче мы использовали счетчик, чтобы вывести цифры от 1 до 5.

Для счетчиков принято использовать буквы **i**, **j** и **k**.

## Цикл for

Цикл **for** является **альтернативой while**. Он более сложен для понимания, но чаще всего его любят больше, чем while за то, что он занимает меньше строчек.

for ( начальные команды; условие окончания цикла; команды после прохода цикла ) {

тело цикла

}

**Начальные команды** - это то, что выполнится перед стартом цикла. Они выполнятся только один раз. Обычно там размещают начальные значения счетчиков, пример: **i = 0**.

**Условие окончания цикла** - пока оно истинное, цикл будет работать, пример: **i <10**.

**Команды после прохода цикла** - это команды, которые будут выполнятся каждый раз при окончании прохода цикла. Обычно там увеличивают счетчики, например: **i++**.

Давайте с помощью цикла **for** выведем последовательно числа от 0 до 9:

/\*

В начале цикла i будет равно нулю,

цикл будет выполнятся пока i < 10,

после каждого прохода к i прибавляется единица:

\*/

for (var i **=** 0; i **<** 10; i**++**) {

alert(i); //выведет 0, 1, 2... 9

}

### **Цикл без тела**

Фигурные скобки в циклах можно не указывать - в этом случае цикл выполнит только одну строку под ним (не рекомендую так делать, часто приводит к ошибкам):

for (var i **=** 0; i **<** 10; i**++**) //<--- точки с запятой нет

alert(i); //выведет 0, 1, 2... 9

А вот если после **)** поставить точку с запятой - цикл закроется и следующая строка в него не попадет, получится так называемый цикл без тела, который в нашем случае просто прокрутится и в результате изменит значение переменной **i**:

for (var i **=** 0; i **<** 10; i**++**); //<--- точка с запятой есть

alert(i); //выведет 9

Такой цикл иногда используется, вы увидите примеры его применения при разборах задач на циклы.

### **Несколько команд в цикле for**

Если нам необходимо выполнить несколько команд в круглых скобках - указываем их через запятую:

for (var i **=** 0, j **=** 2; i **<** 10; i**++**, j**++**, i **=** i **+** j) {

}

Давайте разберем приведенный цикл: до прохода цикла выполнятся две команды: **var i = 0, j = 2** (обратите внимание на то, что var тут пишется один раз), а после каждой итерации - целых три: **i++, j++, i = i + j**.

Этот пример с точки зрения программирования никакой особой пользы не несет, просто схематически показывает, что так можно делать. Запомните его, в дальнейшем это вам пригодится.

## Цикл for для массивов

С помощью цикла **for** можно последовательно перебрать элементы массива. Делается это следующим образом:

var arr **=** [1, 2, 3, 4, 5];

for (var i **=** 0; i **<=** arr.length-1; i**++**) {

alert(arr[i]); //выведет 1, 2, 3, 4, 5

}

Ключевым моментом является то, что мы делаем перебор от нуля до длины массива минус 1 (так как номер последнего элемента массива на единицу меньше его длины).

Можно не отнимать единицу, а место **<=** сделать **<**:

var arr **=** [1, 2, 3, 4, 5];

for (var i **=** 0; i **<** arr.length; i**++**) {

alert(arr[i]); //выведет 1, 2, 3, 4, 5

}

## Цикл for-in

Для перебора объекта используется так называемый цикл **for-in**. Давайте посмотрим, как он работает.

Пусть у нас дан такой объект:

var obj **=** {Коля: 200, Вася: 300, Петя: 400};

Давайте выведем его ключи. Для этого используем такую конструкцию: **for (key in obj)**, где **obj** - это объект, который мы перебираем, а **key** - это переменная, в которую последовательно будут ложится ключи объекта (ее название может быть любым, какое придумаете - такое и будет).

То есть в данном цикле не надо указывать условие окончания - он будет перебирать ключи объекта, пока они не закончатся.

Итак, вот так мы выведем все ключи объекта (по очереди):

var obj **=** {Коля: 200, Вася: 300, Петя: 400};

for (key in obj) {

alert(key); //выведет 'Коля', 'Вася', 'Петя'

}

Если нам нужны не ключи, а значения, нужно обратиться к нашему объекту по ключу, вот так: **obj[key]**.

Как это работает: в переменной **key** сначала будет 'Коля', то есть **obj[key]** в данном случае все равно, что **obj['Коля']**, при следующем проходе цикла в переменной **key** будет 'Вася' и так далее.

Итак, выведем все элементы объекта:

var obj **=** {Коля: 200, Вася: 300, Петя: 400};

for (key in obj) {

alert(obj[key]); //выведет 200, 300, 400

}

## Инструкция break

Иногда нам необходимо прервать выполнение цикла досрочно, в случае с циклом for это значит до того, как цикл переберет все элементы массива.

Зачем такое может понадобится? Например, перед нами стоит задача выводить элементы массива до тех пор, пока не встретится число 3. Как только встретится - цикл должен завершить свою работу.

Такое можно сделать с помощью инструкции **break** - если выполнение цикла дойдет до нее, цикл закончит свою работу.

Давайте решим приведенную выше задачу - оборвем цикл, как только нам встретится число 3:

var arr **=** [1, 2, 3, 4, 5];

for (var i **=** 0; i **<** arr.length; i**++**) {

if (arr[i] **===** 3) {

break; //выходим из цикла

} else {

alert(arr[i]);

}

}

## Инструкция continue

Существует также инструкция **continue**, при достижении которой цикл начинает новую итерацию. Иногда может быть полезна для упрощения кода, хотя практически всегда задачу можно решить и без нее.

**Порядок выполнения работы**

**Примеры решения задач**

### Задача №1

**Задача.** Выведите столбец чисел от **1** до **50**.

**Решение:** воспользуемся циклом while (отделим числа [тегом br](http://old.code.mu/html/br.html) друг от друга, чтобы получить столбец, а не строку):

var i **=** 1;

while (i **<=** 50) {

document.write(i **+** '<br>');

i**++**;

}

Можно также воспользоваться и циклом for:

for (var i **=** 1; i **<=** 50; i**++**) {

document.write(i **+** '<br>');

}

### Задача №2

**Задача.** Дан массив с элементами **[1, 2, 3, 4, 5]**. С помощью цикла **for** выведите все эти элементы на экран.

**Решение:** будем повторять цикл for от 0 до номера последнего элемента массива. Этот номер на единицу меньше количества элементов в массиве, которое можно найти с помощью [свойства length](http://old.code.mu/javascript/array/length.html) таким образом: **arr.length**.

Чтобы цикл прокрутился на единицу меньше длины массива, в условие окончания мы поставим **<**, а не **<=**.

К элементам массива будем обращаться так: **arr[i]**. При этом переменная **i** - это счетчик цикла, который будет меняться от нуля до **arr.length** (не включительно). Таким образом мы последовательно выведем все элементы массива на экран (отделив их [тегом br](http://old.code.mu/html/br.html) друг от друга):

var arr **=** [1, 2, 3, 4, 5];

for (var i **=** 0; i **<** arr.length; i**++**) {

document.write(arr[i] **+** '<br>');

}

### Задача №3

**Задача.** Дан массив с элементами **[2, 3, 4, 5]**. С помощью цикла **for** найдите произведение элементов этого массива.

**Решение:** для таких задач существует **стандартное решение**, которое заключается в том, что циклом **for** перебираются элементы массива и их произведение последовательно записывается в переменную **result** (в ней постепенно накапливается искомое произведение):

var result **=** 1;

var arr **=** [2, 3, 4, 5];

for (var i **=** 0; i **<** arr.length; i**++**) {

result **=** result **\*** arr[i];

}

alert(result);

**Как это работает**: изначально переменная **result** имеет значение **1**, затем при первом проходе цикла в нее записывается ее текущее содержимое (это 1), умноженное на первый элемент массива (это 2). Получится, что в **result** запишется **1\*2** и теперь там будет лежать число 2.

При следующем проходе цикла в **result** запишется текущее значение **result**, умноженное на второй элемент массива (то есть **2\*3=6**). И так далее **пока массив не закончится**.

**Можно переписать** строчку **result = result \* arr[i]** через **\*=** для краткости:

var result **=** 1;

var arr **=** [2, 3, 4, 5];

for (var i **=** 0; i **<** arr.length; i**++**) {

result **\*=** arr[i];

}

alert(result);

### Задача №4 Цикл for-in

**Задача.** Дан объект **obj** с ключами **'Минск', 'Москва', 'Киев'** с элементами **'Беларусь', 'Россия', 'Украина'**. С помощью цикла **for-in** выведите на экран строки такого формата: **'Минск - это Беларусь.'**.

**Решение:** задача не представляет сложности если уметь работать с [циклом for-in](http://old.code.mu/javascript/lang/for.html). Давайте решать задачу поэтапно. Для начала выведем на экран все ключи объекта (это названия городов):

var obj **=** {

'Минск': 'Беларусь',

'Москва': 'Россия',

'Киев': 'Украина'

};

for (var key in obj) {

alert(key);

}

А теперь выведем все значения объекта (это страны):

var obj **=** {

'Минск': 'Беларусь',

'Москва': 'Россия',

'Киев': 'Украина'

};

for (var key in obj) {

alert(obj[key]);

}

Ну, а теперь сформируем строки нужного нам формата:

var obj **=** {

'Минск': 'Беларусь',

'Москва': 'Россия',

'Киев': 'Украина'

};

for (var key in obj) {

alert(key **+** ' - это ' **+** obj[key] **+** '.');

}

**Практическая часть**

### **Циклы while и for**

***Решите эти задачи сначала через цикл while, а затем через цикл for.***

1. Выведите столбец чисел от **1** до **100**.
2. Выведите столбец чисел от **11** до **33**.
3. Выведите столбец четных чисел в промежутке от **0** до **100**.
4. С помощью цикла найдите сумму чисел от **1** до **100**.

### **Работа с for для массивов**

1. Дан массив с элементами **[1, 2, 3, 4, 5]**. С помощью цикла **for** выведите все эти элементы на экран.
2. Дан массив с элементами **[1, 2, 3, 4, 5]**. С помощью цикла **for** найдите сумму элементов этого массива. Запишите ее в переменную **result**.

### **Работа с for-in**

1. Дан объект **obj**. С помощью цикла **for-in** выведите на экран ключи и элементы этого объекта.

var obj **=** {green: 'зеленый', red: 'красный', blue: 'голубой'}

1. Дан объект **obj** с ключами **Коля, Вася, Петя** с элементами **'200', '300', '400'**. С помощью цикла **for-in** выведите на экран строки такого формата: **'Коля - зарплата 200 долларов.'**.

### **Задачи**

1. Дан массив с элементами **2, 5, 9, 15, 0, 4**. С помощью цикла **for** и оператора **if** выведите на экран столбец тех элементов массива, которые больше **3-х**, но меньше **10**.
2. Дан массив с числами. Числа могут быть положительными и отрицательными. Найдите сумму положительных элементов массива.
3. Дан массив с элементами **1, 2, 5, 9, 4, 13, 4, 10**. С помощью цикла **for** и оператора **if** проверьте есть ли в массиве элемент со значением, равным 4. Если есть - выведите на экран 'Есть!' и выйдите из цикла. Если нет - ничего делать не надо.
4. Дан массив числами, например: [10, 20, 30, 50, 235, 3000]. Выведите на экран только те числа из массива, которые начинаются на цифру **1**, **2** или **5**.
5. Дан массив с элементами **1, 2, 3, 4, 5, 6, 7, 8, 9**. С помощью цикла **for** создайте строку **'-1-2-3-4-5-6-7-8-9-'**.
6. Составьте массив дней недели. С помощью цикла **for** выведите все дни недели, а выходные дни выведите жирным.
7. Составьте массив **дней недели**. С помощью цикла **for** выведите все дни недели, а **текущий** день выведите *курсивом*. Текущий день должен храниться в переменной **day**.
8. Дано число **n=1000**. Делите его на **2** столько раз, пока результат деления не станет меньше **50**. Какое число получится? Посчитайте количество итераций, необходимых для этого (*итерация* - это проход цикла), и запишите его в переменную **num**.

**Контрольные вопросы**

1. Какие существуют операторы циклов?
2. Назовите разницу между операторами **while** и **do while**.
3. Для чего нужен оператор **break**?
4. Как можно применять оператор **continue**?