**Лабораторная работа №6**

**Тема работы:** Приемы работы с логическими значениями, циклами и массивами

**Цель работы:** разобрать различные примы работы с логическими значениями, циклами и массивами. Отработать эти приемы на практике

**Теоретические сведения**

**Приемы работы с логическими значениями**

Суть приемов: различные логические операции типа >, <, ==, ===, != могут использоваться не только в ифах, но и сами по себе. Пример:

var a **=** 4;

var b **=** 3;

var c **=** a **>** b; //в переменную c запишется true

Можно также делать комбинации с логическим И && и ИЛИ ||:

var a **=** 3;

var b **=** 4;

var c **=** a **<** 10 **&&** b **<** 10;

**Логические операции без ифов**

Пусть у нас есть функция с ифом. Вот она:

function func(a, b) {

if (a **>** b) {

return true;

} else {

return false;

}

}

Вооружившись знаниями из начала урока, ее можно переписать гораздо проще:

function func(a, b) {

return a **>** b;

}

**Логическое ИЛИ**

Логическое ИЛИ || работает немного не так, как мы ожидаем. Изучите примеры:

var c **=** false **||** 5; //в c запишется 5

var c **=** 5 **||** false; //в c запишется 5

var c **=** 5 **||** 4; //в c запишется 5

var c **=** 4 **||** 5; //в c запишется 4

Вместо false может быть пустая строка, 0, undefined, null, или вообще любое значение, которое в логическом контексте является false.

Таким образом с помощью || можно сделать выбор между двумя значениями. Если одно из них false или приводится к нему - то в переменную будет записано второе.

**Приемы работы с циклами**

**Цикл в цикле**

Циклы, работу с которыми вы уже знаете, можно вкладывать друг в друга. К примеру, давайте решим следующую задачу: выведем на экран строку '111222333444555666777888999'. Тут одним циклом не обойтись - нужно запустить *два вложенных друг в друга цикла*: первый цикл будет перебирать числа (сначала 1, потом 2, потом 3 и так до 9), а второй цикл будет повторять эти числа 3 раза.

Давайте реализуем:

for (var i **=** 1; i **<=** 9; i**++**) {

for (var j **=** 1; j **<=** 3; j**++**) {

document.write(i); //выводит '111', потом '222', потом '333' и так далее

}

}

Обратите внимание: первый цикл имеет счетчик **i**, второй **j**, а если будет еще и третий цикл - то у него счетчик будет переменная **k**. Это стандартные общепринятые названия, следует пользоваться именно ими.

**Пирамидки**

### Строка 'xxxxxxxxxx'

Давайте сделаем строку, заполненную десятью иксами 'x'.

Для этого сделаем переменную **str** и при каждом проходе цикла будем добавлять в нее одну букву 'x'. Сначала в **str** будет '', потом 'x', потом 'xx' и так далее пока цикл не закончится.

После цикла мы получим нужную нам строку:

var str **=** ''; //начальное значение - пустые кавычки

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

str **=** str **+** 'x';

}

alert(str); //выведет 'xxxxxxxxxx'

По сути эта задача напоминает суммирование элементов массива, только вместо чисел мы суммируем строки, постепенно накапливая результат в переменной str.

### Пирамидка с иксами

Давайте теперь выведем на экран следующую пирамидку:

x  
xx  
xxx  
xxxx  
xxxxx

Для этого просто модифицируем предыдущую задачу: при каждом проходе цикла будем выводить на экран содержимое str и <br>.

Так как в цикле **str** постепенно растет, то сначала выведется 'x' и <br>, потом 'xx' и <br> и так далее:

var str **=** '';

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

str **=** str **+** 'x';

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

}

### Строка '123456789'

Давайте сделаем строку '123456789'. Задача аналогична задаче с десятью иксами - только прибавлять будем не иксы, а счетчик цикла:

var str **=** '';

for (var i **=** 1; i **<=** 9; i**++**) {

str **=** str **+** i;

}

alert(str); //выведет '123456789'

### Пирамидка с цифрами

Давайте выведем на экран следующую пирамидку:

111  
222  
333  
444  
555  
666  
777  
888  
999

Похожую задачу мы уже решали, когда выводили строку '111222333444555666777888999'. По сути нам нужна такая же строка, но с <br> в нужных местах. Запускаем два вложенных друг друга цикла:

for (var i **=** 1; i **<=** 9; i**++**) {

for (var j **=** 1; j **<=** 3; j**++**) {

document.write(i);

}

document.write('<br>'); //выводим <br>, чтобы было в столбик

}

### Пирамидка с цифрами

Выведем теперь следующую пирамидку:

1  
22  
333  
4444  
55555  
666666  
7777777  
88888888  
999999999

Задача аналогична предыдущей, только внутренний цикл должен крутиться не 3 раза, а в соответствии с числом: если у нас единица - то повторяем цикл 1 раз, если двойка - два раза, если тройка - три раза.

Как этого достичь: нужно в условии внутреннего цикла написать следующее: **j <= i**. В этом случае если i равен 1, то это все равно, что в условии написать **j <= 1**, а если i равен 2 - то **j <= 2** и так далее:

for (var i **=** 1; i **<=** 9; i**++**) {

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

document.write(i);

}

document.write('<br>');

}

Я думаю, вы уже обратили внимание, что в этой пирамидке мы не используем накопление в строку, которое мы разбирали ранее. Можно и с накоплением, но это получится чуть сложнее:

for (var i **=** 1; i **<=** 9; i**++**) {

var str **=** ''; //каждый раз зачищаем строку

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

str **=** str **+** i;

}

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

}

**While без заданного количества итераций**

Бывают задачи, в которых количество итераций цикла не известно заранее. К примеру: дано число, нужно делить его на 2 столько раз, пока результат не станет меньше 10. Нужно вывести число, которое при этом получится.

В этом случае мы не знаем количество итераций. Не беда - сделаем условием цикла условие *пока число больше 10*:

var num **=** 500;

while (num **>** 10) {

num **=** num **/** 2;

}

alert(num); //после цикла получим результат

Можно использовать и цикл **for** - в этом случае он будет без тела (без {} и в конце ставим точку с запятой):

for (var num **=** 500; num **>** 10; num **=** num **/** 2);

alert(num); //после цикла получим результат

*Если забыть* про *точку с запятой в конце цикла - следующая строчка попадет под цикл и повторится много раз: столько, сколько итераций в цикле.*

**Приемы работы с массивами**

**Заполнение массивов**

Давайте заполним массив десятью иксами 'x'. Для этого воспользуемся методом [push](http://old.code.mu/javascript/array/push.html):

var arr **=** [];

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

arr.push('x');

}

console.log(arr); //выведет ['x', 'x', 'x', 'x', 'x', 'x', 'x', 'x', 'x', 'x']

Давайте теперь заполним массив числами от 1 до 10:

var arr **=** [];

for (var i **=** 1; i **<=** 10; i**++**) {

arr.push(i);

}

console.log(arr); //выведет [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Можно не использовать **push**, а указать ключи напрямую:

var arr **=** [];

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

arr[i] **=** 'x';

}

console.log(arr); //выведет ['x', 'x', 'x', 'x', 'x', 'x', 'x', 'x', 'x', 'x']

var arr **=** [];

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

arr[i] **=** i **+** 1;

}

console.log(arr); //выведет [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

**Переворот массива**

Давайте из массива ['a', 'b', 'c', 'd', 'e'] сделаем ['e', 'd', 'c', 'b', 'a']. Для этого переберем исходный массив с конца и запишем его элементы в новый массив - они будут идти в обратном порядке:

var arr **=** ['a', 'b', 'c', 'd', 'e'];

var result **=** [];

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

result.push(arr[i]);

}

console.log(result); //выведет ['e', 'd', 'c', 'b', 'a']

**Переворот объекта**

Давайте поменяем ключи и значения в объекте, например из {a: 1, b: 2, c: 3, d: 4, e: 5} сделаем {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}.

Для этого переберем циклом **for-in** исходный объект и создадим при этом новый объект **result**. Ключами нового объекта сделаем элементы старого (это obj[key]), а значениями нового объекта - ключи старого (это key):

var obj **=** {a: 1, b: 2, c: 3, d: 4, e: 5};

var result **=** {};

for (var key in obj) {

result[obj[key]] **=** key;

}

console.log(result); //выведет {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}

**Подсчет количества элементов**

Пусть дан массив ['a', 'b', 'c', 'a', 'a', 'b']. Давайте подсчитаем количество одинаковых элементов в этом массиве и сделаем результат в виде объекта {a: 3, b: 2, c: 1}.

Для решения сделаем объект count с начальным значением {a: 0, b: 0, c: 0}. Будем перебирать массив циклом и увеличивать соответствующее значение в объекте count. К примеру: если текущий элемент массива - это 'a', то увеличим count['a'] на единицу - вот так: count['a']++.

Только вместо 'a' следует подставлять текущий элемент массива, вот так: count[arr[i]]++. Давайте напишем окончательный код:

var arr **=** ['a', 'b', 'c', 'a', 'a', 'b'];

var count **=** {a: 0, b: 0, c: 0};

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

count[arr[i]]**++**;

}

console.log(count); //выведет {a: 3, b: 2, c: 1}

Пойдем дальше: то, что объекта **count** имеет изначальное значение {a: 0, b: 0, c: 0} - не очень удобно, ведь мы обычно не знаем, какие элементы есть в массиве.

Пусть объект **count** формируется автоматически таким образом: если такого элемента в count нет, то ему следует присвоить значение 1, а если есть - просто увеличить значение на единицу:

var arr **=** ['a', 'b', 'c', 'a', 'a', 'b'];

var count **=** {};

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

if (count[arr[i]] **===** undefined) {

count[arr[i]] **=** 1;

} else {

count[arr[i]]**++**;

}

}

console.log(count); //выведет {a: 3, b: 2, c: 1}

**Перебор многомерных массивов**

Пусть дан следующий двухмерный массив:

var arr **=** [[1, 2, 3, 4, 5], [6, 7, 8], [9, 10]];

Давайте выведем все его элементы на экран. Для этого нам необходимо запустить два вложенных друг в друга цикла:

var arr **=** [[1, 2, 3, 4, 5], [6, 7, 8], [9, 10]];

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

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

alert(arr[i][j]);

}

}

**Порядок выполнения работы**

**Практическая часть**

**Логические значения**

1. Сделайте функцию, которая параметрами принимает 2 числа. Если эти числа равны - пусть функция вернет true, а если не равны - false.
2. Сделайте функцию, которая параметрами принимает 2 числа. Если их сумма больше 10 - пусть функция вернет true, а если нет - false.
3. Сделайте функцию, которая параметром принимает число и проверяет - отрицательное оно или нет. Если отрицательное - пусть функция вернет true, а если нет - false.

**Циклы**

1. С помощью цикла for сформируйте строку **'123456789'** и запишите ее в переменную **str**.
2. С помощью цикла for сформируйте строку **'987654321'** и запишите ее в переменную **str**.
3. С помощью цикла for сформируйте строку **'-1-2-3-4-5-6-7-8-9-'** и запишите ее в переменную **str**.
4. Нарисуйте пирамиду, как показано на рисунке, только у вашей пирамиды должно быть 20 рядов, а не 5:

x  
xx  
xxx  
xxxx  
xxxxx

1. С помощью двух вложенных циклов нарисуйте следующую пирамидку:

1  
22  
333  
4444  
55555  
666666  
7777777  
88888888  
999999999

1. Нарисуйте пирамиду, как показано на рисунке, воспользовавшись циклом **for**:

xx  
xxxx  
xxxxxx  
xxxxxxxx  
xxxxxxxxxx

**Массивы**

**Заполнение массивов**

1. Заполните массив следующим образом: в первый элемент запишите 'x', во второй 'xx', в третий 'xxx' и так далее.
2. Заполните массив следующим образом: в первый элемент запишите '1', во второй '22', в третий '333' и так далее.
3. Сделайте функцию **arrayFill**, которая будет заполнять массив заданными значениями. Первым параметром функция принимает значение, которым заполнять массив, а вторым - сколько элементов должно быть в массиве. Пример: arrayFill('x', 5) сделает массив ['x', 'x', 'x', 'x', 'x'].
4. Дан массив с числами. Узнайте сколько элементов с начала массива надо сложить, чтобы в сумме получилось больше 10-ти.

**Переворот массива**

1. Дан массив с числами. Не используя метода reverse переверните его элементы в обратном порядке.

**Многомерные массивы**

1. Дан двухмерный массив с числами, например [[1, 2, 3], [4, 5], [6]]. Найдите сумму элементов этого массива. Массив, конечно же, может быть произвольным.
2. Дан трехмерный массив с числами, например [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]. Найдите сумму элементов этого массива. Массив, конечно же, может быть произвольным.