Работа с циклами for и while в JavaScript

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.

http://code.mu/books/javascript/base/rabota-s-ciklami-for-i-while-v-javascript.html