**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Циклические операторы языка JavaScript - For**

|  |
| --- |
| **for**( начальное значение счетчика итераций; условие;приращение счетчика )  {  *//..блок операторов..*  } |

**Важно:** Цикл в Javascript for используется, когда заранее известно, сколько раз должны повториться циклические действия

В качестве начального значения счетчика итераций используется выражение присваивания: например, i=0 - счетчик цикла начинается с нуля.

В качестве приращения счетчика указывается шаг, с которым должен увеличиваться счетчик: например, [i++](http://labs.org.ru/javascript-1/" \l "_javascript" \t "_blank) указывает на то, что каждая итерация цикла будет сопровождаться его увеличением на 1.

Условие цикла - это и есть конечное значение счетчика: например, i<10 - счетчик, достигнув значения 10, останавливает цикл.

**Рассмотрим пример использования цикла for в javascript:**

|  |  |
| --- | --- |
| 1  2  3  4 | **for** (**var** i=0;i<10;i++)  {  document.write(i+"<br>");  } |

В примере на экран выводятся значения счетчика цикла, так как приращение счетчика i++, соответственно на экране будут появляться *0 1 2 3 ... 9*, причем каждая цифра - с новой строки (тег br).

**Задание 1.**  Дано число N и набор из N целых чисел. Вывести в том же порядке все четные числа из данного набора и количество K таких чисел.

**Задание 2.** Даночисло N (N>1) и набор из N вещественных чисел. Набор называется пилообразным, если каждый его внутренний элемент либо больше, либо меньше обоих своих соседей (то есть является «зубцом»). Если данный набор является пилообразным, то вывести 0; в противном случае вывести номер первого элемента, не являющегося зубцом.

**Задание 3.** Даны целые числа K, N и набор из N вещественных чисел: А1, А2, … , АN. Вывести K-е степени чисел из данного набора:

(А1)K, (А2)K, …,(АN)K.

**ОПЕРАТОРЫ ВЫХОДА ИЗ ЦИКЛА *BREAK* И *CONTINUE* В JAVASCRIPT. ОПЕРАТОР *EXIT***

Оператор break прерывает выполнение всего тела цикла, т.е. осуществляет выход из цикла в javaScript.

В то время как оператор continue прерывает выполнение текущей итерации цикла, но продолжая при этом выполнение цикла со следующий итерации.

**Рассмотрим работу операторов break и continue на примере:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **for** (**var** i=0;i<10;i++)  {  **if** (i==4) **continue**;  document.write(i+"<br>");  **if** (i==8) **break**;  } |

В третьей строке примера стоит условие, из-за которого цифра 4 не будет выводиться на экран. В строке №5 осуществляется выход из цикла, но при этом цифра 8 будет выведена на экран, так как оператор вывода стоит до условия (в 4-й строке). Т.е. на экране будет: 0 1 2 3 5 6 7 8 - каждая цифра с новой строки.

**Оператор Exit**

В языке javasctipt предусмотрен оператор выхода из программного кода - оператор **exit.**Чаще всего оператор используется для исключения ошибки ввода пользователя. Рассмотрим пример:

**Пример:** запрашивать пользователя ввести число. Если введено не число, то выводить сообщение "Необходимо число!" и останавливать работу программы

Для выполнения пригодятся функции преобразования типа:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **var** number=prompt("Введите число");  number=parseInt(number); *// возвратит NaN - не число*  x=isNaN(number); *// возвратит true, т.к. не числовое*  **if** (x){  alert("Необходимо число!");  exit; *// выход из программы*  }  alert("Введите второе число");*// при вводе не числа оператор не выполнится*  ... |

Интересная работа с циклом **for** возможна при использовании **одновременно двух счетчиков в цикле**. Рассмотрим пример:

**Пример:** При помощи скрипта распечатать следующие пары переменная - значение в три строки:

*i=0 j=2; i=1 j=3; i=2 j=4*

|  |  |
| --- | --- |
| 1  2  3  4 | **for**(i=0,j=2;i<10,j<5;i++, j++)  {  document.write("<br>i=", i, "j=",j);  } |

Перед выполнением следующего задания рассмотрим пример динамического построения html-страницы при помощи javascript.

**Пример:** необходимо динамически генерировать маркированные и нумерованные списки на веб-странице в зависимости от введенных пользователем данных: запрашивать у пользователя ввести вид списка (нумерованный или маркированный), а затем количество пунктов списка. В зависимости от ответа выводить на экран теги либо маркированного либо нумерованного списка с необходимым количеством пунктов. Если введен несуществующий тип списка, то выдавать сообщение *"Введите правильный тип!"* и осуществлять выход из программы (оператор exit)

**теги маркированного списка:**

|  |
| --- |
| <**ol**>  <**li**></**li**>  <**li**></**li**>  <**li**></**li**>  </**ol**> |

**теги нумерованного списка:**

|  |
| --- |
| <**ul**>  <**li**></**li**>  <**li**></**li**>  <**li**></**li**>  </**ul**> |

В примере для вывода пунктов списка необходимо использовать цикл **[for](http://labs.org.ru/javascript-3/" \l "js3_4" \t "_blank)**. Также понадобятся функции преобразования типов.

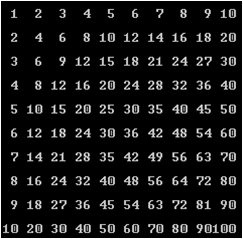
**Показать результат:**

**Задание 4.** Нарисовать шахматную доску 9х9, используя циклы javascript **for**. "Рисовать" доску следует тегами **html** для таблицы:

|  |
| --- |
| <**table** border="1">  <**tr**> *<!-- строка -->*  <**td**></**td**><**td**></**td**> *<!-- две ячейки в строке -->*  </**tr**>  </**table**> |

**Дополнительно:**

1. В ячейки таблицы вывести таблицу умножения, используя счетчики цикла.
2. Первый ряд и первую колонку вывести с красным фоном (атрибут ячейки таблицы bgcolor).

[](http://labs.org.ru/wp-content/uploads/2016/08/1-4.png)

**Циклические операторы языка javaScript - While**

**Синтаксис оператора while:**

|  |
| --- |
| while (условие)  {  *//..блок операторов..*  }; |

**Пример:** Выводить в диалоговое окно степени двойки до 1000

**var** a = 1;

while (a<1000){

a\*=2;

alert(a);

}

**На примере также посмотрим как работают операторы break и continue в цикле while:**

|  |
| --- |
| **var** a = 1;  while (a<1000){  a\*=2;  **if** (a==64)  **continue**;  **if** (a==256)  **break**;  alert(a);  } |

Здесь степени двойки будут выводить до 128 включительно, причем пропущено будет 64. Т.е. в диалоговых окнах мы увидим: *2 4 8 16 32 128*

**Задание 5.** Какие значения выведет следующий фрагмент кода?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **var** counter = 5;  while (counter < 10) {  counter++;  document.write("Counter " + counter);  **break**;  document.write("Эта строка не выполнится.");  } |

**Задание 6.** Выполнить возведение х в степень y, используя цикл while.

**Задание 7**. Дано целое число *N* (> 1). Вывести наибольшее из целых чисел *K*, для которых сумма *1 + 2 + : : : + K* будет меньше или равна *N*, и саму эту сумму.

**Задание 8.** Дано целое число *N* (> 0). Используя операции деления нацело и взятия остатка от деления, вывести все его цифры, начиная с самой правой (разряда единиц).

Задание 9. Дано целое число *N* (> 1). Последовательность чисел Фибоначчи  *FK* определяется следующим образом:

*F*1 = 1, *F*2 = 1, *FK* = *FK-*2 + *FK-*1, *K* = 3, 4, *…*

Проверить, является ли число *N числом Фибоначчи*. Если является, то вывести TRUE, если нет — вывести FALSE.

**Циклические операторы языка javaScript - цикл с постусловием do..while**

|  |
| --- |
| **do**  *{*  *//..блок операторов..*  *}*  **while** (условие); |

**Пример:** Рассмотрим пример использования цикла do while, самостоятельно выясните, что будет выводиться в диалоговое окно:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | **var** a = 1;  **do**  {  a\*=2;  **if** (a==64)  **continue**;  alert(a);  **if** (a==256)  **break**;  }while(a<1000); |

**Задание 10.** Исправьте ошибку в программе, предназначенную для нахождения факториала числа:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | **var** counter = prompt("Введите число");  **var** factorial = 1;  document.write("Факториал числа: " + counter + "! = ");  **do** {  **if** (counter == 0) {  factorial = 1;  **break**;  }  factorial = factorial / counter;  counter = counter + 1;  }  while (counter > 0);  document.write(factorial); |

**Задание 11.** Модифицировать программу про ввод имени пользователем: запрашивать имя пользователя до тех пор, пока пользователь действительно введет имя (т.е. поле будет заполнено и не нажата клавиша cancel)

**Циклические операторы языка javaScript - цикл For In**

Цикл **for in** предназначен для прохода по массивам, коллекциям и объектам:

**Пример:** Распечатать свойства объекта navigator

|  |  |
| --- | --- |
| 1  2  3  4 | **for** (**var** prop **in** navigator)  {  document.write(prop + "<br>");  } |

В примере переменная prop создана для "перебора" всех свойств объекта navigator.

**Оператор обработки исключений в JavaScript - try..catch**

В некоторых случаях код на странице не работает непонятно по какой причине. Где искать ошибку? В таких случаях можно применить оператор try..catch, который пытается выполнить фрагмент кода, и, если в коде есть ошибка, то существует возможность выдать ошибку на экран.

Рассмотрим работу оператора на примере:

**Пример:** написать в программе оператор с ошибкой. Проверять наличие ошибки в предполагаемом ошибочном коде: если ошибка в коде присутствует - выдавать сообщение *"обработка ошибки: название ошибки"*. После проверки ошибочного оператора, независимо от того, есть ли в коде ошибка, выдавать сообщение *"завершающие действия"*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | alert("до");  **try**  {  sdfsdf; *// оператор с ошибкой*  }  **catch**(e)  {  alert("обработка ошибки: "+e.message);  }  **finally**  {  alert("завершающие действия");  }  alert("после"); |

Try с англ. - "пытаться", таком образом, ставим оператор try перед фрагментом кода, который, возможно, содержит ошибку. Если ошибка действительно есть, то оператор catch (с англ. "ловить") сохраняет эту ошибку в объекте e. В дальнейшем ее можно вывести в диалоговое окно - e.message. Если ошибка все же есть, то интерпретатор после ее вывода в нашем примере перейдет на выполнение блока catch, а затем finally (с английского "завершение", "наконец"), который выполнится всегда, независимо от того была ли ошибка или нет. Даже если возникла ошибка в блоке catch.

Блок finally в конструкции необязателен.

**Задание 12.** Выполните пример, описанный выше.

Удалите блок finally и проследите за выполнением кода.

Поставьте вместо ошибочного оператора безошибочный и посмотрите, как будет происходить выполнение кода

**Задание 13.**

Создать игру для двоих:

1. Программа просит ввести число от 1 до 100 первого игрока (второй игрок не видит введенное число). Затем второго игрока просит угадать введенное число. В ответ выводится сообщение «мало» либо «много» в зависимости от введенного ответа. Если игрок угадывает, - выводится поздравление. Если не угадывает – игра продолжается.

2. Просчитывать число попыток и выдавать результат, когда число разгадано.

**Задача 14.**

Начальный вклад в банке равен 10000 руб. Через каждый месяц размер вклада увеличивается на ***P*** процентов от имеющейся суммы (***P*** – вещественное число, 0<***Р***<25). По данному ***P*** определить, через сколько месяцев размер вклада превысит 11000 руб., и вывести найденное количество месяцев ***К*** (целое число) и итоговый размер вклада ***S*** (вещественное число).