## =================== ТЕОРІЯ =======================

## Цикли

Часто постає потреба виконати один і той самий оператор декілька разів. для цього застосовують *оператори циклів*. Цикл складається із *заголовка* і *тіла*. У заголовку циклу зазначається умова завершення циклу, а тіло циклу являє собою оператор, який потрібно виконати декілька разів. Кожне виконання оператора тіла циклу називається його *ітерацією*.

У JavaScript, як і в більшості інших мов програмування, є три типи циклів: з передумовою, післяумовою та параметром.

**Оператор циклу з передумовою** while

Оператор while циклічно виконує оператор (своє тіло) до тих пір, поки умова виконується (логічний вираз приймає значення true).

|  |  |  |
| --- | --- | --- |
| **Програмна структура** | **Аналог на мові блок-схем** | **Приклад.**  Знаходити суму чисел, які вводить користувач, **поки сума є меншою за 100**. |
| while( умова )  оператор ; | **умова**  Оператор  +  – | var sum=0;  while(sum<100)  sum=sum+ +prompt(‘number=’,’0’); |

Якщо тіло циклу складається з більше ніж одного оператора, то необхідно використати складений оператор (записати ці оператори у фігурних дужках).

|  |  |  |
| --- | --- | --- |
| **Програмна структура** | **Аналог на мові блок-схем** | **Приклад.**  Знаходити суму і добуток чисел, які вводить користувач, **поки сума є меншою за 100** |
| while( умова )  **{**  оператор1 ;  . . . . . .  операторN ;  **}** | **умова**  Оператор1  +  –  ОператорN | var sum=0;  var mult=1;  while(sum<=100)  **{**  var number = +prompt(‘number=’,’0’);  sum = sum + number;  mult = mult \* number;  **}** |

Іноді цикл необхідно виконати певну кількість разів. В цьому випадку використовують додаткову змінну *лічильник*, за допомогою якої контролюють кількість ітерацій (кількість разів виконання тіла циклу).

|  |  |
| --- | --- |
| Загальна форма | Приклад. 100 разів вивести на екран слово «Мир». |
| var лічильник = 0;  while( лічильник < кількість повторень )  {  оператор ;  лічильник ++ ;  } | var **i = 0**;  while (**i < 100**)  {  document.write(‘Мир’);    **i++;**  } |

Також цикл часто використовують у випадку, коли деяка величина (*параметр*) повинна змінюватися від деякого початково до кінцевого значення з певним кроком. Зауважимо, що попередній приклад є частковим випадком, коли початкове значення параметра 0, а кінцеве дорівнює кількості виконань.

|  |  |
| --- | --- |
| Загальна форма | Приклад. Вивести на екран усі числа кратні трьом від 6 до 28. |
| var параметр = початкове значення ;  while( параметр <= кінцеве значення)  {  оператор ;  параметр = параметр + крок;  } | var **i = 6**;  while (**i <= 28**)  {  document.write( i );    **i=i+3;**  } |

*Приклад. 4.1.1*  Скласти скрипт зворотного відліку який виводить числа від 5 до 1 а потім повідомлення «БУМ!»

var i=5;

while (i>0){

alert(i--);

}

alert('БУМ!');

**Цикл з післяумовою** do-while

Оператор циклу do-while відрізняється від оператора while тим, що перевірка умови виконується не до, а після виконання тіла цикла (оператора). А тому у операторі циклу do-while тіло циклу виконається принаймні один раз. Як і в циклі з передумовою, тіло виконується поки умова вірна.

|  |  |  |
| --- | --- | --- |
| **Програмна структура** | **Аналог на мові блок-схем** | **Приклад.**  Знаходити суму чисел, які вводить користувач поки не буде введено 0. Зауважимо, що принаймні одне число треба ввести (принаймні один раз тіло циклу потрібно виконати). |
| do  {  оператор ;  }  while ( умова ); | **умова**  Оператор  +  – | var sum=0;  var number;  do  {  number= +prompt(“number=”,“0”);  sum=sum+number;  }  while(number!=0); |

*Приклад. 4.1.2*  Скрипт зворотного відліку:

var i=5;

do {

alert(i--);

}while (i>0);

alert('БУМ!');

*Приклад. 4.2*  Скласти скрипт, який просить від користувача обчислити a+b, де а і b – цілі числа, що генеруються випадковим чином перед кожною спробою.

do {

var a=Math.floor(Math.random()\*100),

b=+(Math.random()\*100).toFixed(0);

var res=+prompt(a+'+'+b+'=','');

} while ((a+b!= res));

alert('Вірно');

**Цикл з параметром for**

Найчастіше у випадку, коли повторення потрібно робити у залежності від зміни деякого параметра або ж у випадку, коли потрібно повторювати тіло циклу деяку кількість разів використовують цикл з параметром for.

Загальний вигляд:

for (<вираз ініціалізації>;<умова продовження>;<ітераційна частина>)

<оператор>;

Аналог на мові блок-схем:

Початкова ініціалізація

Умова продовження циклу

Оператор

Ітераційна частина

+

-

Схематичне зображення виконання оператора:

for(<вираз ініціалізації>;<умова продовження>;<ітераційна частина>)

<оператор>;

**Початок**

Оператор for працює у відповідності до наступного алгоритму:

1. Обчислюється вираз ініціалізації. У цій частині допустима ініціалізація декількох параметрів.
2. Перевіряється умова продовження. Якщо умова невірна, то робота циклу завершується і передається управління наступному оператору.
3. Якщо умова істинна, виконується тіло даного оператора.
4. Виконується приріст одного або декількох параметрів циклу (або виконується довільна інша операція).
5. Здійснюється перехід до кроку 2.

Розв’яжемо раніше розглянуті приклад з використанням циклу for

**Приклад 1.**

|  |  |
| --- | --- |
| Загальна форма | for( лічильник = 0; лічильник < кількість повторень ; лічильник ++)  {  оператор ;  } |
| Приклад, розв’язаний з використанням  for | //----- **100 разів** вивести на екран слово «Мир» ----  for( var **i = 0** ; **i < 100** ; **i++ )**  {  document.write(‘Мир’);  } |
| Розв’язаний з використанням  циклу while | var **i = 0**;  while (**i < 100**)  {  document.write(‘Мир’);  **i++;**  } |

Приклад 2.

|  |  |
| --- | --- |
| Загальна форма | for(**параметр=поч.значення; параметр<=кін.значення; параметр= параметр+крок** )  {  оператор ;  } |
| Приклад, розв’язаний з використанням  for | //------ Вивести на екран усі числа кратні трьом **від 6 до 28**.  for(var **i = 6**; **i <= 28**; **i=i+3 )**  {  document.write( i );  } |
| Розв’язаний з використанням  циклу while | var **i = 6**;  while (**i <= 28**)  {  document.write( i );  **i=i+3;**  } |

Як бачимо, цикл for організовано за принципом циклу while. Тільки запис є значно коротшим (і ініціалізація, і умова продовження, і зміна параметра у одному рядку).

Будь-яка частина for може бути пропущена. Наприклад, можна прибрати початкове значення. Можна прибрати і крок:

for (;умова;) {

//цикл перетворився на аналог while (умова)

}

А можна і взагалі прибрати все, отримавши нескінченний цикл:

for (;;)  // Буде виконуватися вічно

При цьому самі крапки з комою «;» обов'язково повинні бути присутніми, інакше буде помилка синтаксису.

*Приклад. 4.1.3* Скрипт зворотного відліку:

for (var i=5;i>0;i--){

alert(i);

}

alert('БУМ!');

*Приклад. 4.3* Користувач вводить числа поки не натисне відміну. Знайти їх середнє арифметичне

// з передумовою

var s=0,k=0, x=prompt('Введіть число','');

while (x!=null){

s=s+(+x);

k++;

x=prompt('Введіть число','');

}

// частина ідентична для всіх циклів

if (k!=0) {

alert(s/k)}

else{

alert('не введено жодного числа')

}

// з післяумовою

var s=0,k=0;

do {

var x=prompt('Введіть число','');

if (x!=null) {

s+=(+x);

k++;};

} while (x!=null);

// з параметром

for (var s=0,k=0,x=prompt('Введіть число','');x!=null;k++){

s=s+(+x);

x=prompt('Введіть число','');

}

**Опратори break і continue**

Вийти з циклу можна не тільки при перевірці умови але й, взагалі, в будь-який момент. Цю можливість забезпечує оператор break.

Модифікуємо скипт 4.2 так, щоб при натисканні відміни в користувача перепитувало чи він справді хоче вийти.

do {

var a=Math.floor(Math.random()\*100),

b=+(Math.random()\*100).toFixed(0);

var res=prompt(a+'+'+b+'=','');

if (res==null){

if (confirm('Ви справді бажаєте вийти?')){

break};

}

} while ((a+b!= +res));

if (res!=null){ alert('Вірно')};

Буває потрібно вийти одночасно з декількох рівнів циклу. Звичний виклик break не може перервати два цикли відразу. Для цього існують мітки.

Мітка має вигляд ім'я:. ім'я повинне бути унікальним. Вона кладеться перед циклом.:

exit:

for (var i = 0; i < 3; i++) {

for (var j = 0; j < 3; j++) {

var input = prompt(a[ '+i+','+j+’]’, '');  
if (input == null) break exit; // (\*)

}  
 };  
 alert('Готово!');

Директива continue припиняє виконання поточної ітерації циклу та виконує перехід до наступної.. Вона також може застосовуватись з мітками.

*Приклад. 4.4.* Вивеси усі прості числа менші від n;

var n=+prompt('n=','');

document.writeln('<h1>Прості числа до '+n+'</h1><br>');

next: for (var i=1;i<=n;i++){

for (var j=2;j<=Math.sqrt(i);j++){

if (i%j==0){continue next}; //Число не просте, перевіримо наступне.

};

document.writeln(i);

};

### =================== ПРИКЛАДИ ==================

Задача 1. Вивести на екран рядок тексту

а а. . . а с с . . . с

У рядку 37 букв «а» і 10 букв «с»

Задача 1.1. Вивести на екран

000000000000000000000

000000000000000000000

000000000000000000000

000000000000000000000

000000000000000000000

000000000000000000000

000000000000000000000

================================

Задача 2. Вивести на екран таблицю

-------

| | | |

-------

| | | |

-------

| | | |

-------

Задача 3. Вивести на екран

00 00 00 00 00 00 00

-- -- -- -- -- -- --

\/ \/ \/ \/ \/ \/ \/

00 00 00 00 00 00 00

-- -- -- -- -- -- --

\/ \/ \/ \/ \/ \/ \/

00 00 00 00 00 00 00

-- -- -- -- -- -- --

\/ \/ \/ \/ \/ \/ \/

Задача 4. Вивести на екран 10 рядків по 15 букв «о» у кожному. Після кожного рядка вивести рядок із 15 символів «-».

1.ооооооооооооооо

---------------

2.ооооооооооооооо

---------------

3.ооооооооооооооо

---------------

. . . . . . . .

10.ооооооооооооооо

---------------

Задача 5. Вивести на екран

1.1

2.12

3.123

. . .

9.123456789

Задача 6. Вивести на екран

0

00

000

0000

00000

Задача 6. Вивести на екран

0

00

000

0000

00000

0

00

000

0000

00000

0

00

000

0000

00000

Задача 7.1. Вивести на екран

0

000

00000

0000000

0

000

00000

0000000

0

000

00000

0000000

..-------------------------------------------------------------------------------------

### Задачі

1. Користувач вводить числа поки не натисне відміну. Знайти
2. кількість парних;
3. добуток непарних;
4. суму двоцифрових;
5. середнє арифметичне додатних;
6. найбільше від’ємне;
7. кількість більших за перше число;
8. середнє геометричне додатних;
9. найменше з них;
10. кількість трицифрових;
11. добуток кратних 5;

2. Кожне … Вивести початкові та отримані значення.

* 1. двоцифрове натуральне число > 80 збільшити на 7.
  2. двоцифрове натуральне число < 30 збільшити у 5 разів.
  3. натуральне число < 20 збільшити на 10.
  4. двоцифрове натуральне парне число збільшити у 5разів.
  5. двоцифрове натуральне число > 30 збільшити на 10.
  6. натуральне число < 20 збільшити у 9 разів.
  7. двоцифрове натуральне непарне число піднести до квадрата.
  8. трицифрове натуральне число кратне 7 зменшити в 7 разів
  9. двоцифрове натуральне число > 15 збільшити у 9разів
  10. трицифрове натуральне число зменшити на 100;