1. **Цикл For** используется в тех случаях, когда мы можем определить заранее - сколько раз должен выполниться заданный блок кода.

Синтаксис команды выглядит так :

***for (****переменная****=начальное\_значение;***  
*//первым задается начальное значение переменной-счетчика*

*переменная****<= конечное\_значение;***    
*//после этого задается условие, которое проверяется при выполнении каждого цикла*

*переменная****= переменная + шаг)***      
*//и наконец последним задается операция изменения состояния переменной-счетчика*

***{***  
*// исполняемый код*  
***}***

Например:

for (let i = 1; i <= 5; i = i + 1) {

console.log("i = " + i);

}

Результатом выполнения данного цикла будет пятикратное выведение значения переменной i в виде **i = 1, i = 2**и т.д.

**Замечание:**  
Немного забегая вперед, поясним - с помощью + можно объединять строки.  В примере выше, в конструкции "i = " + i мы к строке прибавляем число, которое "приводится" к строковому типу данных и в такой ситуации, если i = 1, получаем в результате строку "i = 1".

**Давайте теперь поподробнее рассмотрим работу цикла FOR.**

for (i = 1; i < 3; i = i + 1) {

console.log("i = " + i);

}

В первой строчке мы задаем параметры работы цикла:

i = 1       //Устанавливаем значение, с которого будет начинаться счет.

i < 3       //Указываем условие, которое будет проверяться каждый цикл (обычно говорят - каждую итерацию)

i = i + 1 //Указываем как будет изменяться счетчик при каждой следующей итерации - увеличиваться на 1

В фигурных скобках мы пишем тело цикла - тот код, который будет выполняться каждое повторение цикла, пока верно наше условие.

{

console.log("i = " + i);  //Тело цикла. В нашем случае - вывод в консоль текста "i =  " и значения переменной i

}

В данном примере происходят следующие действия:

Устанавливаем начальное значение  
1. Переменная i получает значение "1"

Первая итерация  
2. Производится проверка условия что i меньше 3  
3. Поскольку i сейчас равно 1, т.е. меньше 3, то происходит выполнение тела цикла, в нашем примере - вывод на экран переменной i.  
4. Переменная i увеличивается на 1 и становится равной 2.

Вторая итерация  
5. Производится проверка условия что i меньше 3  
6. Поскольку i равно 2, т.е. меньше 3, то происходит выполнение тела цикла  - вывод на экран переменной i.  
7. Переменная i увеличивается на 1 и становится равной 3

Третья итерация  
8. Производится проверка условия что i меньше 3  
9. Поскольку i равно 3, т.е. условие i < 3 перестает быть верным, работа цикла завершается.

У цикла for существует еще одна разновидность - конструкция**for ... in**  
Этой операцией производится перебор всех разрешенных свойств объекта. Более подробно мы рассмотрим использование этой конструкции позже, при изучении объектов, а пока лишь упомянем ее синтаксис :

for (переменная in объект) {

исполняемый код

}

Например, приведенный ниже код выведет все свойства объекта **car**

for (i in car) {

console.log(car[i] + ' ');

}

1. Теперь рассмотрим **цикл while**. В нем происходит выполнение блока кода, пока заданное условие истинно.  
   Синтаксис выглядит следующим образом :

while (выражение, возвращающее логическое значение) {

//в цикле может использоваться, например, оператор сравнения вида x == 10 или просто переменная типа Boolean

исполняемый код

}

Например :

var x = 1;

//объявляем и инициализируем переменную, которую

//мы будем использовать как условие цикла

while (x <= 5) {

console.log(" x= " + x);

x = x + 1;

}

В результате выполнения данного кода произойдет пятикратный вывод значения переменной **х**.

* 1. У **цикла while**есть разновидность : конструкция **do ... while**  
     Данная конструкция отличается от предыдущей тем, что оператор проверки условия расположен ПОСЛЕ основного тела исполняемого кода, что обеспечивает как минимум однократное выполнение блока кода.

Синтаксис выглядит следующим образом:

do {

исполняемый код

} while (переменная == проверочное значение);

Приведем пример использования этой конструкции:

var x = 5;  //установка начального значения

do {    //начало выполнения цикла

console.log(x); //вывод данных в консоль

x = x - 1;           //уменьшение значения на 1

} while (x >= 1);     //проверка условия

В данном случае программа будет выводить в консоль значение переменной "х" в диапазоне от 5 до 1, результат будет "54321".

Ну и завершим урок про циклы рассмотрением двух команд, обеспечивающих **прерывание цикла - break и continue**.

**Ключевое слово break** позволяет нам прервать цикл и перейти к коду, следующему после цикла.  
Рассмотрим пример:

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

if (i == 5) {

break;

}

console.log('i= '+i);

}

В данном случае цикл прервется при достижении переменной i значения 5. До этого момента каждую итерацию будет происходить вывод в консоль значения переменной i.

Ключевое слово **continue** прерывает текущую итерацию цикла и переходит к следующей.  
В качестве примера рассмотрим такой код:

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

  if (i == 5) {

    continue;

  }

  console.log('i= ' + i);

}

В данном случае continue помешает выполниться выводу в консоль при значении переменной равном 5, все остальные значения будут напечатаны.

Говоря о циклах и выходе из них, необходимо также упомянуть такой инструмент, как метки. **Метки в JavaScript** действуют только для циклов и оператора Switch, так что сделать из них полноценный переход в любую точку программы не получится. Однако, этого вполне хватает для перемещения в любое место вложенности внутри циклов.

Например:

var i, j;

metka1:

for (i = 0; i < 5; i = i + 1) {

metka2:

for (j = 0; j < 5; j = j + 1) {

if (i == j) {

continue metka1;

} else {

if (i == 3) {

break metka2;

}

console.log(i, j);

}

}

}

В этом примере в момент, когда значения счетчиков сравняются, произойдет выход из внутреннего цикла и начало следующей итерации внешнего, а при  значении счетчика внешнего цикла, равном значению 3, произойдет прерывание работы внутреннего цикла.

10

20

21

40

41

42

43