Цикл for. JavaScript.

Циклы могут выполнять блоки кода определенное количество раз. Они удобны в случаях, когда вы хотите запустить один и тот же код многократно, добавляя разное значение каждый раз.

JavaScript поддерживает различные виды циклов:
- **for** – перебирает блок кода несколько раз
- **for…in** – перебирает свойства объекта
- **for…of** – перебирает массивы, строки, карты, списки узлов и многое другое
- **while** – проходит через блок кода до тех пор, пока заданное условие истинно
- **do…while** – цикл выполнит блок кода один раз перед проверкой истинности условия, затем цикл будет повторяться до тех пор, пока условие истинно

Чаще всего используется цикл for.

Синтаксис:
```
for (выражение 1; выражение 2; выражение 3) {блок кода, который нужно выполнить}
```

Классический цикл for имеет три компонента, или выражения.
Выражение 1 выполняется до запуска цикла.
Выражение 2 определяет условие для запуска цикла.
Выражение 3 выполняется каждый раз после выполнения цикла. Другими словами, это шаг цикла. Оно не заканчивается точкой с запятой.

Пример 1.

В нижеуказанном примере создается цикл for, который выводит числа от 1 до 5.
```
var i = 1;
for (i=1; i<=5; i++) {
    document.write(i + "<br>");
}
```

Здесь "
" перенос строки. Заставляя текст начинаться с новой строки.
Выражение 1 устанавливает переменную до запуска цикла (i = 1).
Выражение 2 определяет условие для запуска цикла (i должно быть меньше либо равно 5).
Выражение 3 увеличивает значение (i++) после каждого выполнения блока кода в цикле.

Добавляем HTML, чтобы посмотреть на странице сайта.
```
<html>
<body>
<script>
var i = 1;
for (i=1; i<=5; i++) {
    document.write(i + "<br>");
}
</script>
```

```
</body>
</html>
```

Результат:

Пример 2.

Выражение 1 может быть опущено, если значение задано (var i = 1;) до запуска цикла.

```
var i = 1;
for (; i<=5; i++) {
    document.write(i + "<br>");
}
```

Также, вы можете использовать несколько значений в выражении 1, используя **запятые** для их разделения.

```
var i = 1;
for (i=1, text=""; i<=5; i++) {
    text = i;
    document.write(i + "<br>");
}
```

Результат в браузере при добавлении к скрипту HTML для примера 2 будет такой же:

Пример 3.

Теперь давайте выведем 3 раза фразу Learning is fun с добавлением номера, используя цикл for, чтобы выполнять нужную часть кода необходимое количество раз. Можно так же сказать, что на входе в цикл for была одна фраза Learning is fun, а на выходе цикла for три фразы.

```
<html>
<body>
<p id="demo"></p>
<script>
var text = "";
```

```
var i;
for (i = 0; i < 3; i++) {
    text += "Learning is fun " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

Пояснение для оператора присваивания +=

Оператор	Пример	Эквивалентен следующему
+=	x += y	x = x + y

Обратите внимание, что в "Learning is fun " пробел после слова fun. Это позволяет fun не сливаться с числом.

Результат:

Пример 4.

```
var i = 1;
for (k=1; k<10; k++) {
    i += k;
    document.write(i + "<br>");
}
```

Результат:

Пояснение.
1 цикл: i=i+k=1+1=2. После первого цикла i=2.
2 цикл: i=i+k=2+2=4. После второго цикла i=4.
3 цикл: i=i+k=4+3=7. После третьего цикла i=7.
4 цикл: i=i+k=7+4=11. После четвёртого цикла i=11.

5 цикл: i=i+k=11+5=16. После пятого цикла i=16.
6 цикл: i=i+k=16+6=22. После шестого цикла i=22.
7 цикл: i=i+k=22+7=29. После седьмого цикла i=29.
8 цикл: i=i+k=29+8=37. После восьмого цикла i=37.
9 цикл: i=i+k=37+9=46. После девятого цикла i=46.

Пример 5.

Если выражение 2 верно, то цикл запустится ещё раз, если иначе, то цикл закончится. Выражение 2 может быть опущено, но тогда необходимо создать выход (**brek**) внутри самого цикла. Иначе, скрипт не сработает.

Выражение brek, позволяет выйти из цикла и продолжить выполнение кода после цикла.

```
var i = 1;
for (k=1; ; k++) {
    if (k == 5) {brek;}
    i += k;
    document.write(i + "<br>");
}
```

Пояснение для оператора сравнения ==

Оператор	Описание	Пример
==	равно	5 == 10 ложь

Как только k будет равно 5, произойдёт выход из цикла.

Результат:

Пример 6.

Выражение 3 используется для изменения переменной. В нём может выполняться что угодно, а не только увеличение или уменьшение на единицу. Вы можете управлять значением переменной цикла по-разному. Например, можете умножать 'i' на 2 или добавлять к нему фиксированное число. В качестве примера, выведем на страницу сайта чётные числа от 0 до 20, используя цикл for.

```
<html>
<body>
<div id="demo"></div>
<script>
var text = "";
var x = 0;
for (; x <= 20; x += 2) {
```

```
    text += x + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

Здесь document.getElementById("demo").innerHTML = text; изменяет содержимое HTML элемента div с помощью id="demo".

Результат:

Пример 7.

Выражение 3 может быть опущено, если вы изменяет ваши значения внутри цикла.

```
var primer = 9;
for (; primer > -1;) {
    primer--;
    document.write(primer);
}
```

Здесь '-' и '--' дефисы, а не короткие тире и не длинные тире.

Результат:

Пример 8.

Может быть несколько вложенных циклов for. Вот пример скрипта на JavaScript, когда есть внешний цикл for и есть внутренний цикл for.

```
<html>
<head>
    <meta charset="utf-8">
```

```
</head>
<body>
<script>
for (let i = 1; i <= 3; i++) {
  document.write("Внешний цикл, итерация " + i + "." + "<br>");

for (let j = 1; j <= 2; j++) {
  document.write("Внутренний цикл, итерация " + j + "." + "<br>");
 }
}
</script>
</body>
</html>
```

Здесь вместо var используется let внутри скобок for. Такая запись тоже используется. Итерация – повторение какого-либо действия, повторение чего-либо. Например, повторение числовых значений в списке.

Результат вывода будет таким:

Внешний цикл, итерация 1.
Внутренний цикл, итерация 1.
Внутренний цикл, итерация 2.
Внешний цикл, итерация 2.
Внутренний цикл, итерация 1.
Внутренний цикл, итерация 2.
Внешний цикл, итерация 3.
Внутренний цикл, итерация 1.
Внутренний цикл, итерация 2.

В этом примере есть внешний цикл for, который выполняется три раза, и внутри него находится вложенный цикл for, который выполняется два раза на каждую итерацию внешнего цикла. Таким образом, вложенные циклы позволяют повторять определенные операции внутри других операций, что очень полезно при обработке структур данных и выполнении сложных задач.

Пример 9.

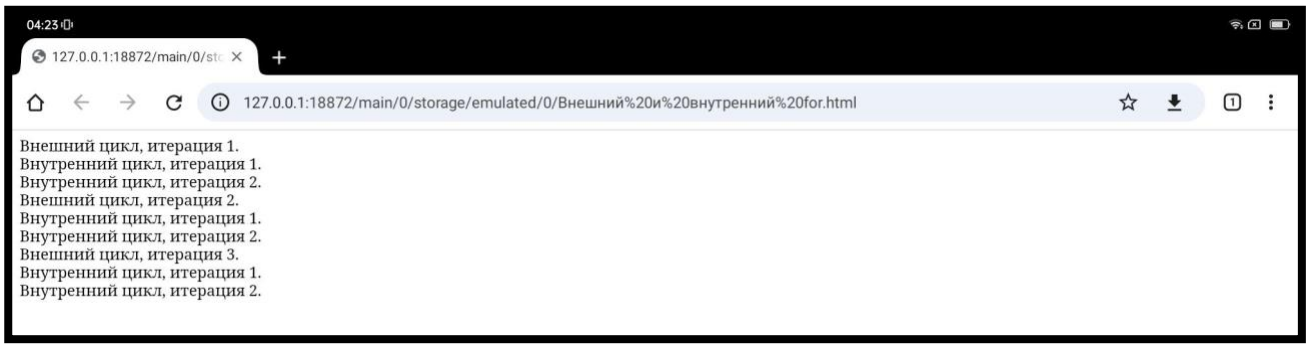
Валидаторы JavaScript не одобряют применение document.write() и пишут "document.write может быть формой eval". Чтобы избежать этого, можно записать пример 8 по-другому.

```
<html>
<head>
        <meta charset="utf-8">
</head>
<body>
<p id="primer"></p>
<script>
let text = "";
for (let i = 1; i <= 3; i++) {
```

```
  text += "Внешний цикл, итерация " + i + "." + "<br>";
for (let j = 1; j <= 2; j++) {
  text += "Внутренний цикл, итерация " + j + "." + "<br>";
 }
}
document.getElementById("primer").innerHTML = text;
</script>
</body>
</html>
```

Здесь валидатор – инструмент для проверки синтаксиса кода JavaScript, поиска ошибок и предупреждений в коде, которые можно исправить.

Результат:



Пример 10.

Вычислим факториал числа 5, используя цикл for.
Здесь факториал числа n – произведение всех чисел от 1 до n. Факториал не может быть отрицательным или дробным, то есть в факториале перемножаются числа, возникающие естественным образом при счёте (1, 2, 3, 4, 5, 6, 7 и так далее). Обозначается факториал числа 5 как 5!

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

```
<html>
<head>
        <meta charset="utf-8">
</head>
<body>
<div id="demo"></div>
<script>
let i;
let factorial = 1;
```

```
for (i = 2; i <= 5; i++) {
   factorial *= i;
}
document.getElementById("demo").innerHTML = "Факториал числа 5 равен " +
factorial;
</script>
</body>
</html>
```

Здесь "фокус" в ***=**. Пояснение для оператора присваивания *=

Оператор	Пример	Эквивалентен следующему
*=	x *= y	x = x * y

factorial умножается на текущее значение i при каждой итерации. В конце скрипта будет выведен результат этого умножения, а не каждая строка цикла.

Результат:

Аналогично для оператора присваивания /= , то есть деления. Будет выведен результат деления, а не каждая строка цикла.

Пример 11.

Обычно цикл for используется для повторения значений в списке.

```
<html>
<body>
<script>
let arr = [1, 2, 3];
for (let k = 0; k < arr.length; k++) {
   document.write(arr[k] + "<br>");
}
</script>
</body>
</html>
```

Здесь [ ] – обозначение массива. Массив хранит множество значений в одной переменной. Элементы массива маркируются с 0. Первый элемент массива [0], второй [1].
arr – название переменной. Выбирается произвольно.
arr.lenght – объект arr со свойством lenght. Используется синтаксис с точкой. length подсчитывает число символов в свойстве или в строке. В примере массив. Свойство

массива lenght возвращает число его элементов. Если массив пустой, то свойство lenght возвращает 0.
Что такое объект JavaScript?
Переменные являются контейнерами для значений. Объекты также являются переменными, но они могут хранить много значений.

Результат:

Пример 12 заключительный.

Код на JavaScript, который использует цикл for для вывода месяцев года на экран в браузере.

```
<html>
<head>
        <meta charset="utf-8">
</head>
<body>
<script>
var months = ['Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь', 'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь', 'Декабрь'];
for (var i = 0; i < months.length; i++) {
   document.write(months[i] + '<br>');
}
</script>
</body>
</html>
```

Этот код создает массив months, содержащий названия месяцев. Затем цикл for перебирает каждый элемент массива и выводит его на экран браузера с помощью document.write(). Каждый месяц выводится на новой строке с использованием тега
, что мы делали неоднократно выше.

Результат:

Примечание.
Писал я эту статью на планшете с операционной системой Android. Примеры для просмотра в браузере. Заходил в файловый менеджер, находил нужный файл html и открывал его в браузере. Дело в том, что не все файловые менеджеры, прошитые в Android или скаченные из Google Play, пригодны для этого. Из многих не откроешь правильно html. Рекомендую использовать приложение с таким ярлыком

Первые около 10 минут при открытии html, менеджер работает в режиме сервера. Об этом есть уведомление в строке уведомлений планшета. В эти 10 минут файл html будет отображаться правильно. Если время вышло, нужно заходить заново и открывать заново.