22.04.2021

Здравствуйте уважаемые обучающиеся
Учебная дисциплина: Технологии публикации
Тема урока: циклы JavaScript(продолжение)

Вам необходимо самостоятельно изучить текст лекции, выполнить задания и письменно ответить на контрольные вопросы.

Выполненную работу оформить и отправить отдельным файлом (электронный документ) в личное сообщение через социальные сети VK. или на почту torincrust@gmail.com

Если такой возможности нет, выполненное задание предоставить в распечатанном (рукописном) виде после возобновления занятий.

1. КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ:

4. Цикл do...while

Цикл `do...while;` проверяет условие продолжения после выполнения цикла. В отличие от цикла `while`, в `do...while;` тело цикла выполняется как минимум один раз, так как условие проверяется в конце цикла, а не в начале. Данный цикл используется реже, чем `while`, так как на практике ситуация, когда требуется хотя бы однократное исполнение цикла, встречается редко.

```
var result = '';
var i = 0;
do {
  i += 1;
  result += i + ' ';
} while (i < 5);
document.write(result);
```

JavaScript

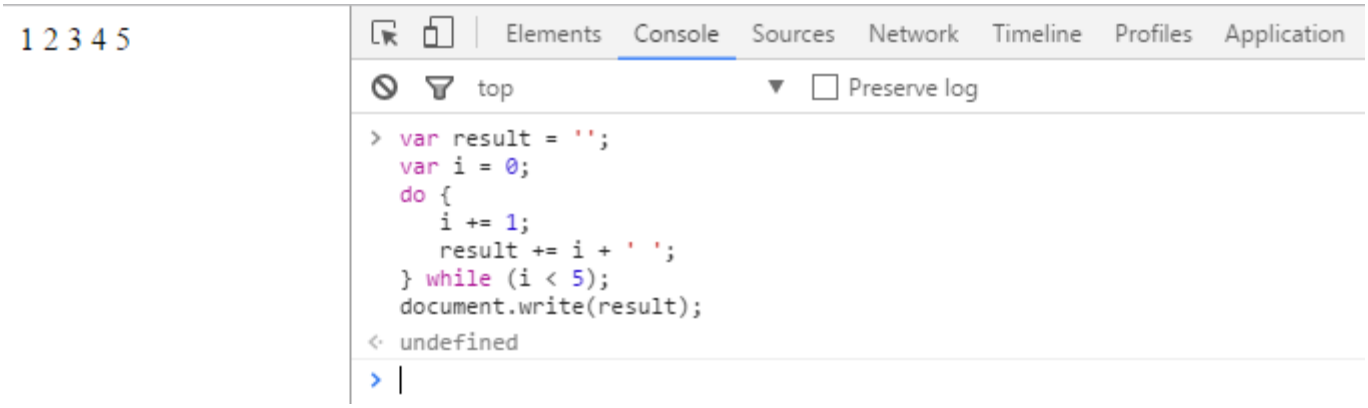


РИС. 6. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЦИКЛА DO...WHILE

В следующем примере операторы внутри цикла выполняются один раз, даже если условие не выполняется.

```
var i = 10;
do {
   document.write(i + ' ');
   i++;
} while (i < 10);
```

JavaScript

5. Бесконечные циклы

При создании любого цикла можно создать бесконечный цикл, который никогда не завершится. Такой цикл может потенциально продолжать работать до тех пор, пока работает компьютер пользователя. Большинство современных браузеров могут обнаружить это и предложат пользователю остановить выполнение скрипта. Чтобы избежать создания бесконечного цикла, вы должны быть уверены, что заданное условие в какой-то момент вернёт `false`. Например, следующий цикл задаёт условие, которое никогда не возвращает ложь, так как переменная `i` никогда не будет меньше `10`:

```
for (var i = 25; i > 10; i++) {
document.write("Это предложение будет выводиться бесконечно...<br>");
}
```

JavaScript

6. Вложенные циклы

Цикл внутри другого цикла называется **вложенным**. При каждой итерации цикла вложенный цикл выполняется полностью. Вложенные циклы можно создавать с помощью цикла `for` и цикла `while`.

```
for (var count = 1; count < 3; count++) {
  document.write(count + ". Строка цикла<br>");
  for (var nestcount = 1; nestcount< 3; nestcount++) {
    document.write("Строка вложенного цикла<br>");
  }
}
```

JavaScript

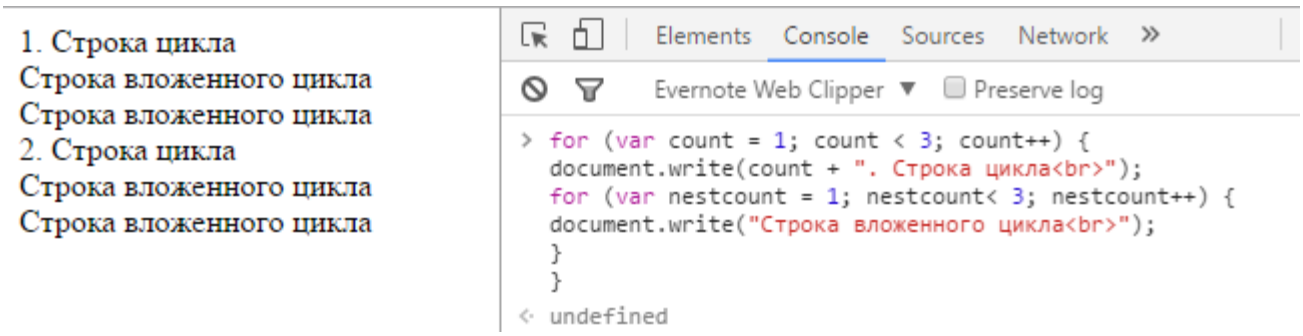


РИС. 7. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ВЛОЖЕННОГО ЦИКЛА FOR

7. Управление циклом

Циклом можно управлять с помощью операторов `break;` и `continue;`.

7.1. Оператор break;

Оператор `break;` завершает выполнение текущего цикла. Он используется в исключительных случаях, когда цикл не может выполняться по какой-то причине, например, если приложение обнаруживает ошибку. Чаще всего оператор `break;` является частью конструкции `if`.

Когда оператор `break;` используется без метки, он позволяет выйти из цикла или из инструкции `switch`. В следующем примере создаётся счётчик, значения которого должны изменяться от `1` до `99`, однако оператор `break` прерывает цикл после 14 итераций.

```
for (var i = 1; i < 100; i++) {
  if (i == 15) {
    break;
    }
  document.write(i);
  document.write(' <br>');
}
```

JavaScript

РИС. 8. РЕЗУЛЬТАТ РАБОТЫ ОПЕРАТОРА BREAK В ЦИКЛЕ FOR

Для вложенных циклов оператор `break;` используется с меткой, с помощью которой завершается работа именованной инструкции. Метка позволяет выйти из любого блока кода. Именованной инструкцией может быть любая инструкция, внешняя по отношению к оператору `break;`. В качестве метки может быть имя инструкции `if` или имя блока инструкций, заключенных в фигурные скобки только для присвоения метки этому блоку. Между ключевым словом `break;` и именем метки не допускается перевод строки.

```JavaScript
outerloop:
  for(var i = 0; i < 10; i++) {
    innerloop:
      for(var j = 0; j < 10; j++) {
        if (j > 3) break; // Выход из самого внутреннего цикла
        if (i == 2) break innerloop; // То же самое
        if (i == 4) break outerloop; // Выход из внешнего цикла
        document.write("i = " + i + " j = " + j + "<br>");
      }
    }
document.write("FINAL i = " + i + " j = " + j + "<br>");
```

JavaScript

7.2. Оператор continue;

Оператор `continue;` останавливает текущую итерацию цикла и запускает новую итерацию. При этом, цикл `while` возвращается непосредственно к своему условию, а цикл `for` сначала вычисляет выражение инкремента, а затем возвращается к условию.
В этом примере на экран будут выведены все чётные числа:

```
var i;
 for(i = 1; i <= 10; i++) {
  if (i % 2 !== 0) {
   continue;
  }
  document.write("<br><b>чётное число</b> = " + i);
 }
```

JavaScript

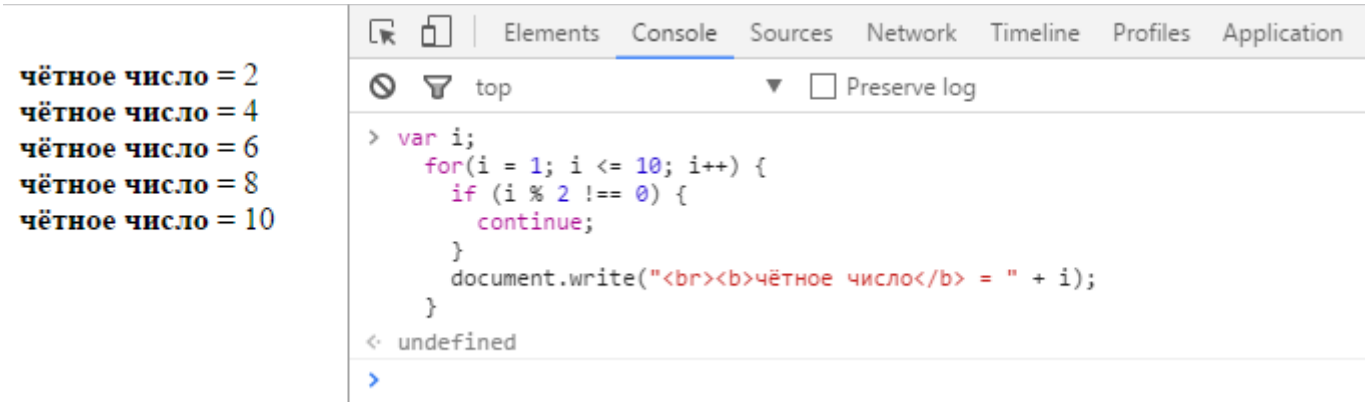
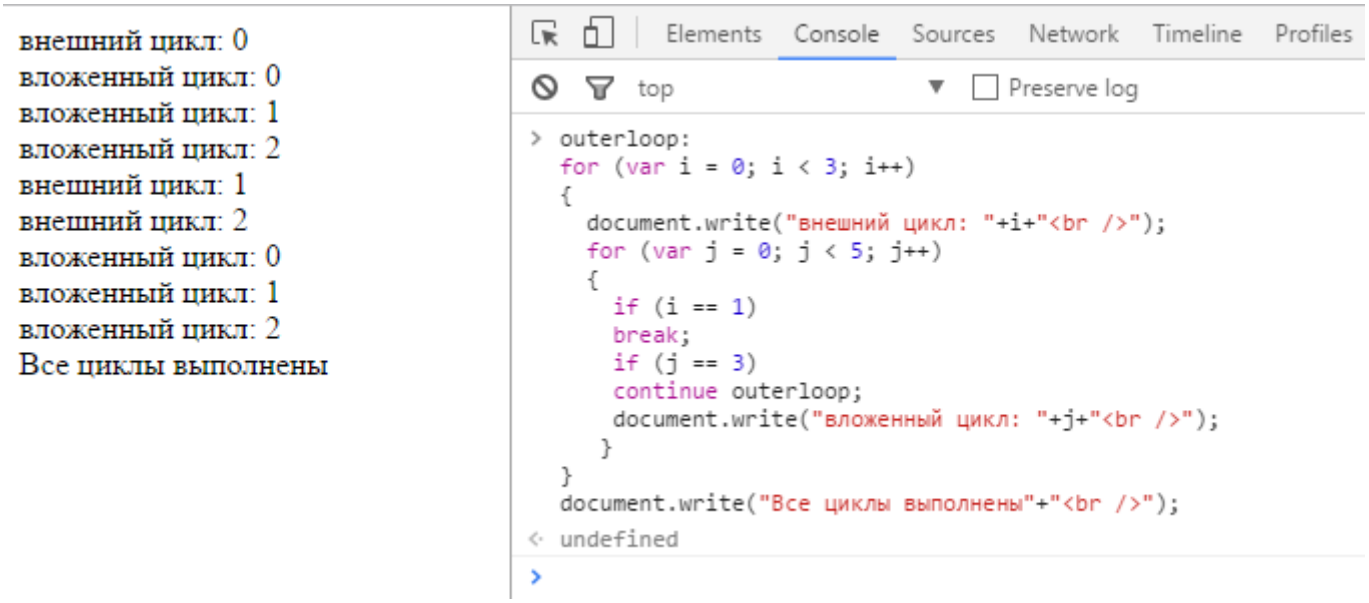


РИС. 9. РЕЗУЛЬТАТ РАБОТЫ ОПЕРАТОРА CONTINUE В ЦИКЛЕ FOR

Оператор `continue;` также может применяться во вложенных циклах с меткой.

```
outerloop:
for (var i = 0; i < 3; i++)
{
 document.write("внешний цикл: "+i+"");
 for (var j = 0; j < 5; j++)
 {
  if (i == 1)
  break;
  if (j == 3)
  continue outerloop;
  document.write("вложенный цикл: "+j+"");
 }
}
document.write("Все циклы выполнены"+"");
```

JavaScript



2. ЗАДАНИЯ К ЛЕКЦИИ

Прочитайте текст лекции, напишите краткий опорный конспект по данному материалу и ответьте на контрольные вопросы.

3. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Способы управления циклами?
2. Что такое бесконечный цикл?
3. Что из себя представляет цикл do while?