

Раздел

[Основы JavaScript](#)

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Язык программирования JavaScript](#)
[→ Основы JavaScript](#) 1-го октября 2020

Циклы while и for

При написании скриптов зачастую встаёт задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода предусмотрены *циклы*.

Цикл «while»

Цикл while имеет следующий синтаксис:

```
1 while (condition) {  
2     // код  
3     // также называемый "телом цикла"  
4 }
```

Код из тела цикла выполняется, пока условие condition истинно.

Например, цикл ниже выводит i, пока i < 3:

```
1 let i = 0;  
2 while (i < 3) { // выводит 0, затем 1, затем 2  
3     alert( i );  
4     i++;  
5 }
```

Одно выполнение тела цикла по-научному называется *итерация*. Цикл в примере выше совершает три итерации.

Если бы строка i++ отсутствовала в примере выше, то цикл бы повторялся (в теории) вечно. На практике, конечно, браузер не позволит такому случиться, он предоставит пользователю возможность остановить «подвисший» скрипт, а JavaScript на стороне сервера придётся «убить» процесс.

Любое выражение или переменная может быть условием цикла, а не только сравнение: условие while вычисляется и преобразуется в логическое значение.

Например, while (i) – более краткий вариант while (i != 0):

```
1 let i = 3;  
2 while (i) { // когда i будет равно 0, условие станет ло:  
3     alert( i );  
4     i--;  
5 }
```

Фигурные скобки не требуются для тела цикла из одной строки

Если тело цикла состоит лишь из одной инструкции, мы можем опустить фигурные скобки {...}:

```
1 let i = 3;  
2 while (i) alert(i--);
```

Раздел

Основы JavaScript

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей
итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



Редактировать на GitHub



Цикл «do...while»

Проверку условия можно разместить под телом цикла, используя специальный синтаксис `do...while`:

```
1 do {
2   // тело цикла
3 } while (condition);
```

Цикл сначала выполнит тело, а затем проверит условие `condition`, и пока его значение равно `true`, он будет выполняться снова и снова.

Например:

```
1 let i = 0;
2 do {
3   alert( i );
4   i++;
5 } while (i < 3);
```



Такая форма синтаксиса оправдана, если вы хотите, чтобы тело цикла выполнилось **хотя бы один раз**, даже если условие окажется ложным. На практике чаще используется форма с предусловием: `while(...) {...}`.

Цикл «for»

Более сложный, но при этом самый распространённый цикл — цикл `for`.

Выглядит он так:

```
1 for (начало; условие; шаг) {
2   // ... тело цикла ...
3 }
```



Давайте разберёмся, что означает каждая часть, на примере. Цикл ниже выполняет `alert(i)` для `i` от 0 до (но не включая) 3:

```
1 for (let i = 0; i < 3; i++) { // выведет 0, затем 1, за
2   alert(i);
3 }
```



Рассмотрим конструкцию `for` подробнее:

часть		
начало	<code>i = 0</code>	Выполняется один раз при входе в цикл
условие	<code>i < 3</code>	Проверяется <i>перед</i> каждой итерацией цикла. Если оно вычислится в <code>false</code> , цикл остановится.
шаг	<code>i++</code>	Выполняется <i>после</i> тела цикла на каждой итерации <i>перед</i> проверкой условия.
тело	<code>alert(i)</code>	Выполняется снова и снова, пока условие вычисляется в <code>true</code> .

В целом, алгоритм работы цикла выглядит следующим образом:

```
1 Выполнить *начало*
2 → (Если *условие* == true → Выполнить *тело*, Выполнит
3 → (Если *условие* == true → Выполнить *тело*, Выполнит
4 → (Если *условие* == true → Выполнить *тело*, Выполнит
5 → ...
```

То есть, *начало* выполняется один раз, а затем каждая итерация заключается в проверке *условия*, после которой выполняется *тело* и *шаг*.

Если тема циклов для вас нова, может быть полезным вернуться к примеру выше и воспроизвести его работу на листе бумаги, шаг за шагом.

Вот в точности то, что происходит в нашем случае:

```
1 // for (let i = 0; i < 3; i++) alert(i)
2
3 // Выполнить начало
4 let i = 0;
5 // Если условие == true → Выполнить тело, Выполнить шаг
6 if (i < 3) { alert(i); i++ }
7 // Если условие == true → Выполнить тело, Выполнить шаг
8 if (i < 3) { alert(i); i++ }
9 // Если условие == true → Выполнить тело, Выполнить шаг
10 if (i < 3) { alert(i); i++ }
11 // ...конец, потому что теперь i == 3
```

Встроенное объявление переменной

В примере переменная счётчика `i` была объявлена прямо в цикле. Это так называемое «встроенное» объявление переменной. Такие переменные существуют только внутри цикла.

```
1 for (let i = 0; i < 3; i++) {
2   alert(i); // 0, 1, 2
3 }
4 alert(i); // ошибка, нет такой переменной
```

Вместо объявления новой переменной мы можем использовать уже существующую:

```
1 let i = 0;
2
3 for (i = 0; i < 3; i++) { // используем существующую
4   alert(i); // 0, 1, 2
5 }
6
7 alert(i); // 3, переменная доступна, т.к. была объявлена
```

Пропуск частей «for»

Любая часть `for` может быть пропущена.

Для примера, мы можем пропустить `начало` если нам ничего не нужно делать перед стартом цикла.

Вот так:

```
1 let i = 0; // мы уже имеем объявленную i с присвоенным значением
2
3 for (; i < 3; i++) { // нет необходимости в "начале"
4   alert(i); // 0, 1, 2
5 }
```

Можно убрать и `шаг` :

```
1 let i = 0;
2
3 for (; i < 3;) {
4   alert(i++);
5 }
```

Раздел

[Основы JavaScript](#)

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей итерации: `continue`

Метки для `break/continue`

Итого

Задачи (7)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Раздел

[Основы JavaScript](#)

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей
итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Это сделает цикл аналогичным `while (i < 3)`.

А можно и вообще убрать всё, получив бесконечный цикл:

```
1 for (;;) {
2     // будет выполняться вечно
3 }
```

При этом сами точки с запятой `;` обязательно должны присутствовать, иначе будет ошибка синтаксиса.

Прерывание цикла: «break»

Обычно цикл завершается при вычислении условия в `false`.

Но мы можем выйти из цикла в любой момент с помощью специальной директивы `break`.

Например, следующий код подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит, а затем – выдаёт:

```
1 let sum = 0;
2
3 while (true) {
4
5     let value = +prompt("Введите число", '');
6
7     if (!value) break; // (*)
8
9     sum += value;
10
11 }
12 alert( 'Сумма: ' + sum );
```

Директива `break` в строке `(*)` полностью прекращает выполнение цикла и передаёт управление на строку за его телом, то есть на `alert`.

Вообще, сочетание «бесконечный цикл + `break`» – отличная штука для тех ситуаций, когда условие, по которому нужно прерваться, находится не в начале или конце цикла, а посередине.

Переход к следующей итерации: continue

Директива `continue` – «облегчённая версия» `break`. При её выполнении цикл не прерывается, а переходит к следующей итерации (если условие всё ещё равно `true`).

Её используют, если понятно, что на текущем повторе цикла делать больше нечего.

Например, цикл ниже использует `continue`, чтобы выводить только нечётные значения:

```
1 for (let i = 0; i < 10; i++) {
2
3     // если true, пропустить оставшуюся часть тела цикла
4     if (i % 2 == 0) continue;
5
6     alert(i); // 1, затем 3, 5, 7, 9
7 }
```

Для чётных значений `i`, директива `continue` прекращает выполнение тела цикла и передаёт управление на следующую итерацию `for` (со следующим числом). Таким образом `alert` вызывается только для нечётных значений.

Раздел

Основы JavaScript

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей
итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



Редактировать на GitHub



Директива continue позволяет избежать вложенности

Цикл, который обрабатывает только нечётные значения, мог бы выглядеть так:

```
1 for (let i = 0; i < 10; i++) {
2
3   if (i % 2) {
4     alert( i );
5   }
6
7 }
```

С технической точки зрения он полностью идентичен.

Действительно, вместо continue можно просто завернуть действия в блок if.

Однако мы получили дополнительный уровень вложенности фигурных скобок. Если код внутри if более длинный, то это ухудшает читаемость, в отличие от варианта с continue.

Нельзя использовать break/continue справа от оператора „?“

Обратите внимание, что эти синтаксические конструкции не являются выражениями и не могут быть использованы с тернарным оператором ?. В частности, использование таких директив, как break/continue, вызовет ошибку.

Например, если мы возьмём этот код:

```
1 if (i > 5) {
2   alert(i);
3 } else {
4   continue;
5 }
```

...и перепишем его, используя вопросительный знак:

```
1 (i > 5) ? alert(i) : continue; // continue здесь г
```

...то будет синтаксическая ошибка.

Это ещё один повод не использовать оператор вопросительного знака ? вместо if.

Метки для break/continue

Бывает, нужно выйти одновременно из нескольких уровней цикла сразу.

Например, в коде ниже мы проходимся циклами по i и j, запрашивая с помощью prompt координаты (i, j) с (0,0) до (2,2):

```
1 for (let i = 0; i < 3; i++) {
2
3   for (let j = 0; j < 3; j++) {
4
5     let input = prompt(`Значение на координатах (${i},${j})`);
6
7     // Что если мы захотим перейти к Готово (ниже) сразу?
8
9   }
10 }
11
```

```
alert('Готово!');
```

Раздел

Основы JavaScript

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



Редактировать на GitHub



Нам нужен способ остановить выполнение если пользователь отменит ввод.

Обычный break после input лишь прервёт внутренний цикл, но этого недостаточно. Достичь желаемого поведения можно с помощью меток.

Метка имеет вид идентификатора с двоеточием перед циклом:

```
1  labelName: for (...) {
2    ...
3  }
```

Вызов break <labelName> в цикле ниже ищет ближайший внешний цикл с такой меткой и переходит в его конец.

```
1  outer: for (let i = 0; i < 3; i++) {
2
3    for (let j = 0; j < 3; j++) {
4
5        let input = prompt(`Значение на координатах (${i},${j})`);
6
7        // если пустая строка или Отмена, то выйти из обоих
8        if (!input) break outer; // (*)
9
10       // сделать что-нибудь со значениями...
11    }
12 }
13
14 alert('Готово!');
```



В примере выше это означает, что вызовом break outer будет разорван внешний цикл до метки с именем outer, и управление перейдёт со строки, помеченной (*), к alert('Готово!').

Можно размещать метку на отдельной строке:

```
1  outer:
2  for (let i = 0; i < 3; i++) { ... }
```

Директива continue также может быть использована с меткой. В этом случае управление перейдёт на следующую итерацию цикла с меткой.



Метки не позволяют «прыгнуть» куда угодно

Метки не дают возможности передавать управление в произвольное место кода.

Например, нет возможности сделать следующее:

```
1  break label; // не прыгает к метке ниже
2
3  label: for (...)
```

Вызов break/continue возможен только внутри цикла, и метка должна находиться где-то выше этой директивы.

Итого

Мы рассмотрели 3 вида циклов:

- while – Проверяет условие перед каждой итерацией.



Раздел

[Основы JavaScript](#)

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей
итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



[Редактировать на GitHub](#)



- `do...while` – Проверяет условие после каждой итерации.
- `for (;;)` – Проверяет условие перед каждой итерацией, есть возможность задать дополнительные настройки.

Чтобы организовать бесконечный цикл, используют конструкцию `while (true)`. При этом он, как и любой другой цикл, может быть прерван директивой `break`.

Если на данной итерации цикла делать больше ничего не надо, но полностью прекращать цикл не следует – используют директиву `continue`.

Обе этих директивы поддерживают метки, которые ставятся перед циклом. Метки – единственный способ для `break/continue` выйти за пределы текущего цикла, повлиять на выполнение внешнего.

Заметим, что метки не позволяют прыгнуть в произвольное место кода, в JavaScript нет такой возможности.

✓ Задачи

Последнее значение цикла

важность: 3

Какое последнее значение выведет этот код? Почему?

```
1 let i = 3;
2
3 while (i) {
4   alert( i-- );
5 }
```

решение



Какие значения выведет цикл while?

важность: 4

Для каждого цикла запишите, какие значения он выведет. Потом сравните с ответом.

Оба цикла выводят `alert` с одинаковыми значениями или нет?

1.

Префиксный вариант `++i`:

```
1 let i = 0;
2 while (++i < 5) alert( i );
```

2.

Постфиксный вариант `i++`

```
1 let i = 0;
2 while (i++ < 5) alert( i );
```

решение

Какие значения выведет цикл for?

важность: 4

Для каждого цикла запишите, какие значения он выведет. Потом сравните с ответом.

Оба цикла выведут `alert` с одинаковыми значениями или нет?

Раздел

[Основы JavaScript](#)

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей
итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



[Редактировать на GitHub](#)



1.

Постфиксная форма:

```
1 for (let i = 0; i < 5; i++) alert( i );
```

2.

Префиксная форма:

```
1 for (let i = 0; i < 5; ++i) alert( i );
```

решение

Выведите чётные числа

важность: 5

При помощи цикла `for` выведите чётные числа от 2 до 10.

[Запустить демо](#)

решение

Замените `for` на `while`

важность: 5

Перепишите код, заменив цикл `for` на `while`, без изменения поведения цикла.

```
1 for (let i = 0; i < 3; i++) {  
2   alert( `number ${i}!` );  
3 }
```

решение

Повторять цикл, пока ввод неверен

важность: 5

Напишите цикл, который предлагает `prompt` ввести число, большее 100. Если посетитель ввёл другое число – попросить ввести ещё раз, и так далее.

Цикл должен спрашивать число пока либо посетитель не введёт число, большее 100, либо не нажмёт кнопку Отмена (ESC).

Предполагается, что посетитель вводит только числа. Предусматривать обработку нечисловых строк в этой задаче необязательно.

[Запустить демо](#)

решение

Вывести простые числа

важность: 3

Натуральное число, большее 1, называется **простым**, если оно ни на что не делится, кроме себя и 1.

Другими словами, $n > 1$ – простое, если при его делении на любое число кроме 1 и n есть остаток.

Например, 5 это простое число, оно не может быть разделено без остатка на 2, 3 и 4.

Раздел

[Основы JavaScript](#)

Навигация по уроку

Цикл «while»

Цикл «do...while»

Цикл «for»

Прерывание цикла: «break»

Переход к следующей
итерации: continue

Метки для break/continue

Итого

Задачи (7)

Комментарии

Поделиться



[Редактировать на GitHub](#)

Напишите код, который выводит все простые числа из интервала от 2 до n .

Для n = 10 результат должен быть 2,3,5,7 .

P.S. Код также должен легко модифицироваться для любых других интервалов.



решение

Проводим [курсы по JavaScript и фреймворкам](#).



Комментарии

перед тем как писать...

© 2007—2020 Илья Кантор | [о проекте](#) | [связаться с нами](#) | [пользовательское соглашение](#) | [политика конфи](#)

