Директивы break и continue

Выход: break

Выйти из цикла можно не только при проверке условия но и, вообще, в любой момент. Эту возможность обеспечивает директива `break`.

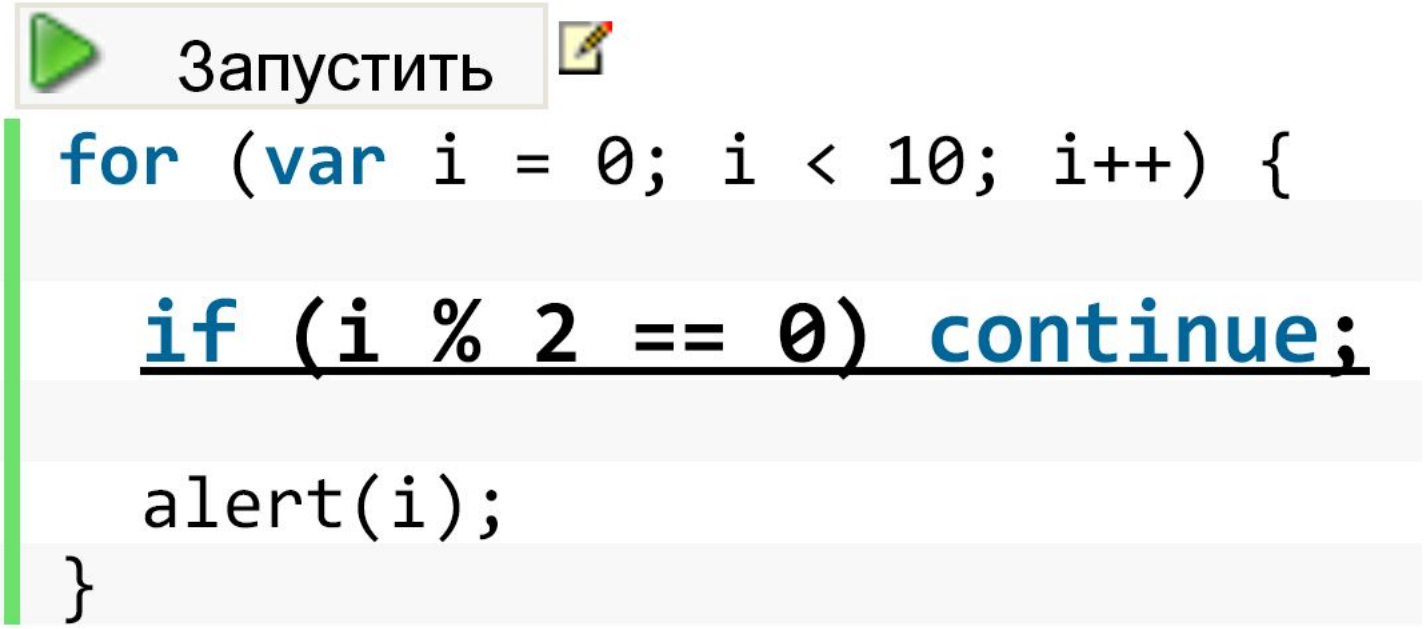
Например, бесконечный цикл в примере прекратит выполнение при `i==5`:

```
01 var i=0;
02
03 while(1) {
04   i++;
05
06   if (i==5) break;
07
08   alert(i);
09 }
10
11 alert('Последняя i = '+ i ); // 5 (*)
```

Выполнение продолжится со строки `(*)`, следующей за циклом.

Следующая итерация: continue

Директива `continue` прекращает выполнение *текущей итерации* цикла. Например, цикл ниже не выводит четные значения:



Для четных `i` срабатывает `continue`, выполнение блока прекращается и управление передается на `for`.

Нельзя использовать break/continue справа от оператора ‘?’

Обычно мы можем заменить `if` на оператор вопросительный знак `'?'`.

То есть, запись:

```
if (условие) {
  a();
} else {
  b();
}
```

..Аналогична записи:

```
условие ? a() : b();
```

В обоих случаях в зависимости от условия выполняется либо `a()` либо `b()`.

Но разница состоит в том, что оператор вопросительный знак `'?'`, использованный во второй записи, возвращает значение.

Синтаксические конструкции, которые не возвращают значений, нельзя использовать в операторе `'?'`. К таким относятся большинство конструкций и, в частности, `break/continue`.

Поэтому такой код приведёт к ошибке:

```
(i > 5) ? alert(i) : continue;
```

Метки

Бывает нужно выйти одновременно из нескольких уровней цикла.

Представим, что нужно ввести значения точек. У каждой точки есть две координаты `(i, j)`. Цикл ввода значений `i,j = 0..2` может выглядеть так:

Запустить

```
01 for (var i = 0; i < 3; i++) {
02
03   for (var j = 0; j < 3; j++) {
04
05     var input = prompt("Значение в координатах " + i + "," + j, "");
06
07     if (input == null) break; // (*)
08
09   }
10 }
11 alert('Готово!');
```

Здесь `break` используется, чтобы прервать ввод, если посетитель нажал на `Отмена`. Но обычный вызов `break` в строке `(*)` не может прервать два цикла сразу. Как же прервать ввод полностью? Один из способов — поставить *метку*.

Метка имеет вид `"имя:"`, имя должно быть уникальным. Она ставится перед циклом, вот так:

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

Можно также выносить ее на отдельную строку. Вызов `break outer` прерывает управление цикла с такой меткой, вот так:

Запустить

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

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

    var input = prompt('Значение в координатах '+i+','+j, '');

    if (input == null) break outer; // (*)

  }
}
alert('Готово!');
```

Директива `continue` также может быть использована с меткой. Управление перепрыгнет на следующую итерацию цикла с меткой.

Метки можно ставить в том числе на блок, без цикла:

Запустить

```
01 my: {
02
03   for (;;) {
04     for (i=0; i<10; i++) {
05       if (i>4) break my;
06     }
07   }
08
09   some_code; // произвольный участок кода
10
11 }
12 alert("После my"); // (*)
```

В примере выше, `break` перепрыгнет через `some_code`, выполнение продолжится сразу после блока `my`, со строки `(*)`. Возможность ставить метку на блоке используется редко. Обычно метки ставятся перед циклом.

Goto?

В некоторых языках программирования есть оператор `goto`, который может передавать управление на любой участок программы.

Операторы `break/continue` более ограниченны. Они работают только внутри циклов, и метка должна быть не где угодно, а выше по уровню вложенности.

В JavaScript нет `goto`.

Натуральное число, большее 1, называется *простым*, если оно ни на что не делится, кроме себя и 1. Важность: 4

Другими словами, `n>1` - простое, если при делении на любое число от `2` до `n-1` есть остаток.

Создайте код, который выводит все простые числа из интервала от `2` до `10`.
Результат должен быть: `2,3,5,7`.

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

Решение

Схема решения

```
Для всех i от 1 до 10 {
 проверить, делится ли число i на какое-либо из чисел до него
   если делится, то это i не подходит, берем следующее
   если не делится, то i - простое число
}
```

```
<!DOCTYPE HTML>
<html>
<head>
  <!-- Тег meta для указания кодировки -->
  <meta charset="utf-8">
</head>
<body>

  <script>
    var test = 5;
    alert(test);
  </script>

</body>
</html>
```