Циклы

Цикл «while»:

```
while (condition) {
  // код
  // также называемый "телом цикла"
}
```

Код из тела цикла выполняется, пока условие *condition* истинно. Например, цикл ниже выводит **i**, пока **i < 3**:

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

Если бы строка **i++** отсутствовала в примере выше, то цикл бы повторялся (в теории) вечно.

Цикл «while»:

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)**:

```
let i = 3;
while (i) { // когда i будет равно 0, условие станет ложным, и цикл остановится
        alert( i );
        i--;
}
```

Цикл «do…while»:

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

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

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

Такая форма синтаксиса оправдана, если вы хотите, чтобы тело цикла **выполнилось хотя бы один раз**, даже если условие окажется ложным.

Цикл «for»:

Более сложный, но при этом самый распространённый цикл.

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

Цикл ниже выполняет alert(i) для i от 0 до (но не включая) 3:

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

Цикл «for»:

Рассмотрим конструкцию **for** подробней:

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

В целом, алгоритм работы цикла выглядит следующим образом:

*Выполнить *начало**
→ *(Если *условие* == true → Выполнить *тело*, Выполнить *шаг*)*
→ *(Если *условие* == true → Выполнить *тело*, Выполнить *шаг*)*
→ *(Если *условие* == true → Выполнить *тело*, Выполнить *шаг*)*
→ *...*

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:

```
let sum = 0;

while (true) {
  let value = +prompt('Введите число', ' ');
  if (!value) {
        break;
  }
  sum += value;
}

alert( 'Сумма: ' + sum );
```

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

Переход к следующей итерации: continue:

Директива **continue** – «облегчённая версия» **break**. При её выполнении цикл не прерывается, а переходит к следующей итерации (если условие все 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:

```
for (let i = 0; i < 10; i++) {

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

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.