Циклы JavaScript обеспечивают многократное выполнение повторяющихся вычислений. Они оптимизируют процесс написания кода, выполняя одну и ту же инструкцию или блок инструкций, образующих тело цикла, заданное число раз (используя переменную-счётчик) или пока заданное условие истинно. Циклы выполняют обход последовательности значений. Однократное выполнение цикла называется итерацией.

На производительность цикла влияют количество итераций и количество операций, выполняемых в теле цикла каждой итерации.

В JavaScript существуют следующие операторы цикла:

1) for используется когда вы заранее знаете, сколько раз вам нужно что-то сделать;

2) for...in используется для обхода свойств объектов;

3) while используется когда вы не знаете, сколько раз нужно что-то сделать;

4) do...while работает аналогично с оператором while. Отличается тем, что do...while всегда выполняет выражение в фигурных скобках, по крайней мере один раз, даже если проверка условия возвращает false.

1. Цикл for

Цикл for используется для выполнения итераций по элементам массивов или объектов, напоминающих массивы, таких как arguments и HTMLCollection. Условие проверяется перед каждой итерацией цикла. В случае успешной проверки выполняется код внутри цикла, в противном случае код внутри цикла не выполняется и программа продолжает работу с первой строки, следующей непосредственно после цикла.

Следующий цикл выведет на консоль строчку Hello, JavaScript! пять раз.

for (var i = 0; i < 5; i++) {

console.log(i + ": Hello, JavaScript!");

}

JavaScript

РИС. 1. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЦИКЛА FOR НА КОНСОЛИ

1.1. Как работает цикл for

Цикл for состоит из трёх разных операций:

Шаг 1. инициализация var i = 0; — объявление переменной-счётчика, которая будет проверяться во время выполнения цикла. Эта переменная инициализируется со значением 0. Чаще всего в качестве счётчиков цикла выступают переменные с именами i, j и k.

Шаг 2. проверка условия i < 5; — условное выражение, если оно возвращает true, тело цикла (инструкция в фигурных скобках) будет выполнено. В данном примере проверка условия идёт до тех пор, пока значение счётчика меньше 5.

Шаг 3. завершающая операция i++ — операция приращения счётчика, увеличивает значение переменной var i на единицу. Вместо операции инкремента также может использоваться операция декремента.

По завершении цикла в переменной var i сохраняется значение 1. Следующий виток цикла выполняется для for (var i = 1; i < 5; i++) { }. Условное выражение вычисляется снова, чтобы проверить, является ли значение счётчика i всё ещё меньше 5. Если это так, операторы в теле цикла выполняются ещё раз. Завершающая операция снова увеличивает значение переменной на единицу. Шаги 2 и 3 повторяются до тех пор, пока условие i < 5; возвращает true.

1.2. Вывод значений массива

Чтобы вывести значения массива с помощью цикла for, нужно задействовать свойство массива length. Это поможет определить количество элементов в массиве и выполнить цикл такое же количество раз.

Приведённый ниже скрипт выведет на экран пять сообщений с названиями цветов:

var flowers = ["Rose", "Lily", "Tulip", "Jasmine", "Orchid"];

for (var i = 0; i < flowers.length; i++){

alert(flowers[i] + " - это цветок.");

}

JavaScript

Если значение свойства length не изменяется в ходе выполнения цикла, можно сохранить его в локальной переменной, а затем использовать эту переменную в условном выражении. Таким образом можно повысить скорость выполнения цикла, так как значение свойства length будет извлекаться всего один раз за всё время работы цикла.

var flowers = ["Rose", "Lily", "Tulip", "Jasmine", "Orchid"], len = flowers.length;

for (var i = 0; i < len; i++){

alert(flowers[i] + " - это цветок.");

}

JavaScript

2. Цикл for...in

Циклы for...in используются для обхода свойств объектов, не являющихся массивами. Такой обход также называется перечислением. При обходе рекомендуется использовать метод hasOwnProperty(), чтобы отфильтровать свойства, которые были унаследованы от прототипа.

Для примера создадим объект с помощью литерала объекта.

var user = {

name: 'Alice',

age: 25,

country: 'Russia'

};

for (var prop in user) {

console.log(prop + ": " + user[prop]);

}

JavaScript

РИС. 2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЦИКЛА FOR...IN НА КОНСОЛИ

Предположим, что в сценарии до или после создания объекта user прототип объекта Object был расширен дополнительным методом clone().

if (typeof Object.prototype.clone === 'undefined') {

Object.prototype.clone = function () {};

}

JavaScript

Так как цепочка наследования прототипа постоянно проверяется интерпретатором, то все объекты автоматически получают доступ к новому методу.

РИС. 3. РЕЗУЛЬТАТ ПОВТОРНОГО ВЫПОЛНЕНИЯ ЦИКЛА FOR...IN НА КОНСОЛИ

Чтобы избежать обнаружения этого метода в процессе перечисления свойств объекта user, используется метод hasOwnProperty(), который отфильтрует свойства прототипа.

var user = {

name: 'Alice',

age: 25,

country: 'Russia'

};

if (typeof Object.prototype.clone === 'undefined') {

Object.prototype.clone = function () {};

}

for (var prop in user) {

if (user.hasOwnProperty(prop)) {

console.log(prop + ": " + user[prop]);

}

}

JavaScript

РИС. 4. РЕЗУЛЬТАТ ПЕРЕЧИСЛЕНИЯ СВОЙСТВ ОБЪЕКТА С ПОМОЩЬЮ МЕТОДА HASOWNPROPERTY()

3. Цикл while

Цикл while - цикл с предварительной проверкой условного выражения. Инструкция внутри цикла (блок кода в фигурных скобках) будет выполняться в случае, если условное выражение вычисляется в true. Если первая проверка даст результат false, блок инструкций не выполнится ни разу.

После завершения итерации цикла условное выражение опять проверяется на истинность и процесс будет повторяться до тех пор, пока выражение не будет вычислено как false. В этом случае программа продолжит работу с первой строки, следующей непосредственно после цикла (если таковая имеется).

Данный цикл выведет на экран таблицу умножения для числа 3:

var i = 1;

var msg = '';

while (i < 10) {

msg+= i + ' x 3 = ' + (i \* 3) + '<br>';

i++;

}

document.write(msg);

JavaScript

РИС. 5. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЦИКЛА WHILE

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

nested-loop

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.

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