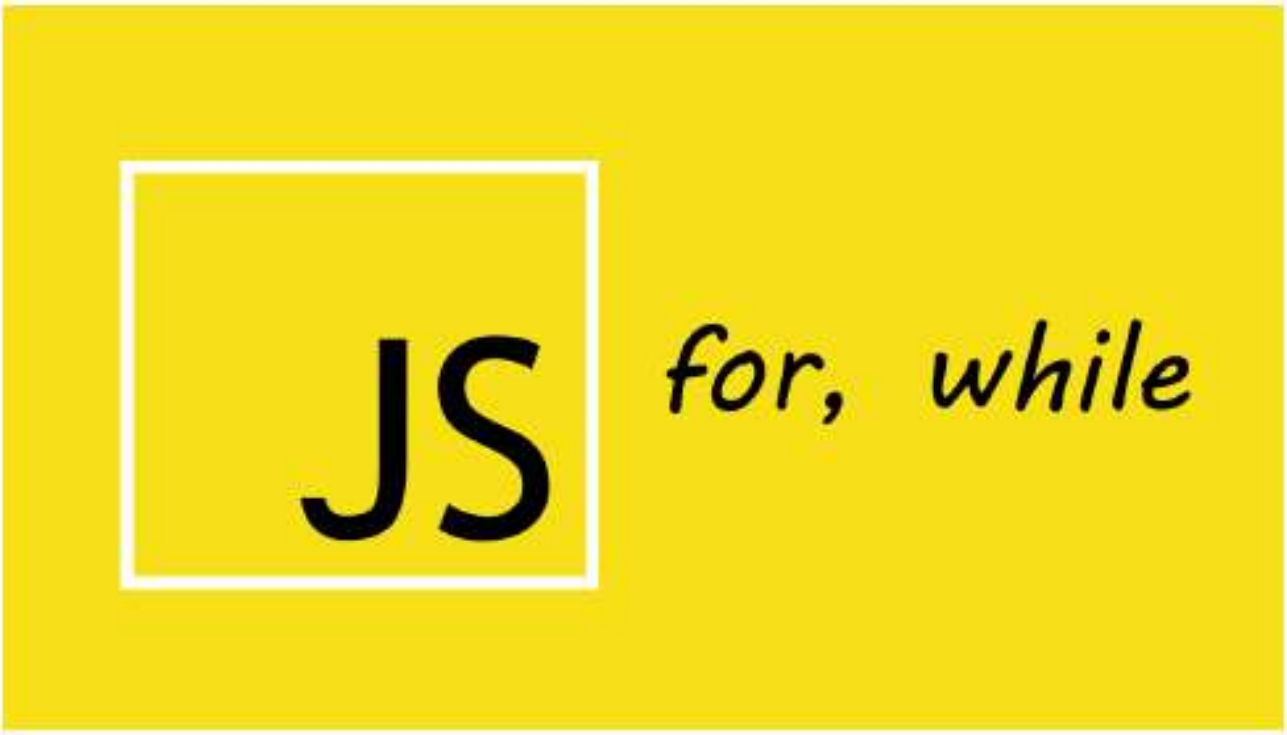
Лабораторная работа №7. Циклы в JavaScript



Содержание:

Назначение и виды циклов

Циклы – это простой способ для многократного выполнения одних и тех же действий (кода).

При этом однократное выполнения кода в цикле называется **итерацией**.

В JavaScript существуют различные виды циклов, но все они, по сути, делают одно и тоже. Просто с помощью одних циклов более просто решаются одни задачи, с помощью других – иные:

- for;
- while;
- do...while;
- for...in;

- for...of (появился в версии ES6).

Цикл for

Данный цикл в основном используется, когда известно точное количество повторений. Этот цикл ещё называют циклом со счётчиком.

Синтаксис цикла «for»:

```
for (инициализация; условие; финальное выражение) {
  /* тело цикла */
}
```

Основные части конструкции цикла «for»:

- **инициализация** - это выражение, которое выполняется один раз перед выполнением цикла; обычно используется для инициализации счётчика;
- **условие** - это выражение, истинность которого проверяется перед каждой итерацией; если выражение **вычисляется как истина**, то выполняется итерация; в противном случае цикл «for» завершает работу;
- **финальное выражение** - это выражение, которое выполняется в конце каждой итерации; обычно используется для изменения счетчика;
- **тело цикла** - инструкции, выполнение которых нужно повторять.

Рассмотрим пример цикла, который выведет в консоль числа от 1 до 8:

```
// цикл «for» от 1 до 8, с шагом 1
for (var i = 1; i <= 8; i++) {
  console.log(i);
}
```

В этом примере:

- инициализация: `var i = 1` (объявление переменной `i` и присвоение ей значения 1);
- условие выполнения цикла: `i <= 8` (пока значение переменной `i` меньше или равно 8);

- финальное выражение, которое нужно выполнять в конце каждой итерации: `i++` (увеличение значение переменной `i` на 1);
- инструкция, которую нужно выполнять: `console.log(i)` (выведение значения счётчика в консоль).

При этом если тело цикла состоит из одной инструкции, то её можно **не заключать в фигурные скобки**.

Таким образом, пример, приведённый выше, можно записать ещё так:

```
// цикл «for» от 1 до 8, с шагом 1
for (var i = 1; i <= 8; i++) console.log(i);
```

Необязательные части цикла цикла «for».

В «for» все части цикла являются не обязательными.

Например, можно пропустить выражение инициализации:

```
var i = 1;
// цикл «for»
for (; i <= 8; i++) {
  console.log(i);
}
```

В этом случае инициализацию переменной можно вынести за пределы цикла.

Условие в «for» тоже является не обязательным. Без условия цикл будет выполняться бесконечное количество раз. В этом случае чтобы его прервать (выйти из цикла) необходимо использовать инструкцию `break`.

```
// цикл «for»
for (var i = 1; ; i++) {
  if (i >= 8) { // условие прерывания цикла
    break;
  }
  console.log(i);
}
```

Финальное выражение в «for» также является не обязательным. Счётчик цикла в этом случае можно, например, изменять в теле.

```
// цикл «for»
for (var i = 1; i <= 8; ) {
  console.log(i);
  i++; // увеличение счетчика на 1
}
```

В «for» можно вообще опустить 3 выражения (бесконечный цикл):

```
var i = 1;
// цикл «for»
for (;;) {
  if (i >= 8) {
    break;
```

```
  }
  console.log(i);
  i++;
}
```

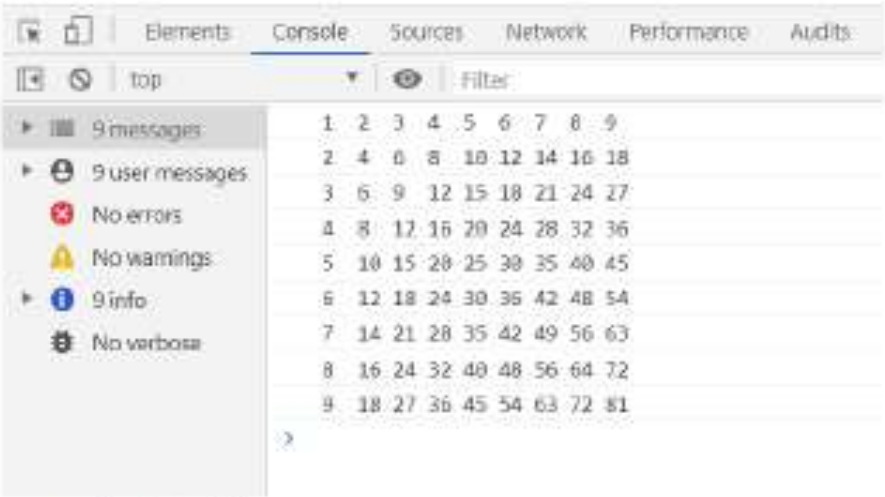
Пример использования цикла «for» для перебора элементов массива:

```
var arr = ["a", "b", "c"]; // массив
for (var i = 0, length = arr.length; i < length; i++) {
  console.log(arr[i]);
}
```

Написать программу, которая выводит сумму элементов в массиве

Написать программу, которая выводит максимальный элемент массива

Пример, в котором выведем таблицу умножения в консоль. Для реализации этого примера будем использовать вложенные циклы.




```
var output = '';
for (var i = 1; i <= 9; i++) {
  for (var j = 1; j <= 9; j++) {
    output += ' ' + i * j;
    if (i * j < 10) {
      output += ' ';
    }
  }
  console.log(output);
  output = '';
}
```

Цикл называется вложенным, если он находится в теле другого цикла.

Цикл while

Данный цикл предназначен для многократного выполнения одних и тех же инструкций до тех пор, пока истинно некоторое условие. Цикл «while» в основном используется, когда количество повторений заранее не известно.

```
while (условие) {
  /* тело цикла */
}
```

Истинность условия проверяется перед каждым выполнением. Если перед первой итерацией условие ложно, то цикл не выполнится ни разу.

Пример, в котором выведем в консоль чётные числа в диапазоне от 1 до 8:

```
// объявим переменную a и присвоим ей значение 0
let a = 0;
//цикл while с условием a <= 8
while (a <= 8) {
  // увеличим значение переменной a на 1
  a++;
  // если число нечётное (остаток от деления на 2 не равен 0), то...
  if (a % 2 !== 0) {
    // пропустим дальнейшее выполнение текущей итерации и перейдём к следующей
    continue;
  }
  // выведем значение переменной a в консоль
  console.log(a);
}
```

Цикл do...while

Цикл «do...while», также как и цикл «while», выполняет одни и те же инструкции до тех пор, пока указанное условие истинно. Но в отличие от «while» в «do...while» условие проверяется после выполнения инструкций. Поэтому цикл «do...while» в любом случае выполнится не меньше одного раза, даже если условие изначально ложно.


```
do {
  /* тело цикла */
} while (условие)
```

Пример, в котором выведем в консоль сумму чисел, которые будем запрашивать у пользователя с помощью функции `prompt`:

```
// num – переменная для хранения числа, введённого пользователем
// sum – переменная для хранения суммы чисел
let num, sum = 0;
// цикл «do...while»
do {
  // запросим у пользователя данные и приведём их к числу
  num = +prompt ('Введите число', '');
  // если то, что ввёл пользователь после приведения является числом, то...
  if (num) {
    // прибавим к сумме число, введённое пользователем
    sum += num;
  }
// если num приводится к истине, то выполняем ещё итерацию
} while (num);

console.log(sum);
```

Цикл for...in

Цикл «for...in» предназначен для перебора перечисляемых имён свойств объекта. В JavaScript свойство является перечисляемым, если его внутренний флаг `[[Enumerable]]` равен `true`.

Свойства объекта, которые не относятся к перечисляемым, в цикле не участвуют.

Например, объект (массив) созданный с использованием функции-конструктора `Array` или его литеральной записи имеет не перечисляемые свойства от `Array.prototype` и `Object.prototype`, такие как `indexOf()`, `some()`, `toString()` и др. Они не будут участвовать в цикле.

```
/* цикл для перебора всех перечисляемых свойств объекта
    - key – переменная, в которую будет помещаться имя свойства объекта
    - object – объект, свойства которого нужно перебрать */
for (key in object) {
```

```
  /* тело цикла */
}
```

Переберём свойства объекта, созданного с помощью литеральной записи:

```
let car = {
  manufacturer: 'Ford',
  model: 'Fiesta',
  color: 'black'
};
for (let propName in car) {
  // propName - имя свойства
  // car[propName] - значение свойства
  console.log(propName + ' = ' + car[propName]);
}
// в консоль будет выведено: manufacturer = Ford, model = Fiesta, color = black
```

Кроме этого, следует отметить, что цикл `for...in` проходит не только по перечисляемых свойствам этого объекта, но и по наследуемым.

```
let item = {
  a: 1,
  b: 2
}
let newItem = Object.create(item);
newItem.c = 3;
newItem.d = 4;
for (let propName in newItem) {
  console.log(propName);
}
// в консоли будет выведено: c, d, a, b
```

Если вам наследуемые свойства не нужно учитывать, то их можно пропустить:

```
for (let propName in newItem) {
  // переходим к следующей итерации, если текущее свойство не принадлежит этому объекту
  if(!newItem.hasOwnProperty(propName)) {
    continue;
  }
  console.log(propName);
}
// в консоли будет выведено: c, d
```

Использование цикла for... in для перебора массива. В массиве свойствами являются числовые индексы.

```
// массив
var arr = ["Rock", "Jazz", "Classical", "Hip Hop"];
// перебор массива с помощью цикла for in
for (let index in arr) {
  // index - индекс элемента массива
  // arr[index] - значение элемента
  console.log(arr[index]);
}
// в результате в консоль будет выведено: "Rock", "Jazz", "Classical", "Hip Hop"
```

Цикл for...in проходит по свойствам в произвольном порядке. Поэтому если при переборе массива для вас важен порядок символов, то данный цикл лучше не использовать.

При использовании цикла for...in стоит обратить внимание на то, что если вы к массиву добавили свои пользовательские свойства, то он по ним тоже пройдётся:

```
var arr = [5, 7, -3];
arr.sum = 2;
for (var key in arr) {
  console.log(arr[key]);
}
// в консоль будет выведено 5, 7, -3, 2
```

Если вам такой сценарий не нужен, то тогда для перебора массивов лучше использовать обычный цикл for.

Использование цикла for...in для перебора символов в строке:

```
var str = 'Метод';
for (var key in str) {
  console.log(str[key]);
}
// М, е, т, о, д
```

Инструкции break и continue

Внутри тела цикла можно использовать специальные инструкции: `break` и `continue`.

Инструкция **«break» предназначена для прекращения выполнения текущего цикла**. Другими словами, она осуществляет выход и передачу управления инструкции, идущей после этого цикла.

Пример, в котором завершим цикл по перебору элементов массива, если его текущий элемент не будет являться числом:

```
// массив
var arr = [5, 3, "a", 4, "b", 16];
// цикл «for» для перебора массива arr
for (var i = 0, length = arr.length; i < length; i++) {
  // если текущий элемент массива не является числом, то...
  if (typeof arr[i] !== 'number') {
    // прерываем выполнение цикла
    break;
  }
  // выводим текущий элемент массива в консоль
  console.log(arr[i]);
}
// в результате в консоль будет выведено: 5, 3
```

Инструкция **«continue» предназначена для прекращения дальнейшего выполнения кода и перехода к следующей итерации цикла**.

Пример, в котором найдём в слове «программирование» символы «а» и «о», и выведем их позиции в консоль:

```
// строка
var str = 'программирование';
// цикл "for" для перебора символов строки
for (var i = 0, length = str.length; i < length; i++) {
  // если текущий символ не равен а и о, то...
  if (str[i] !== 'а' && str[i] !== 'о') {
    // прекращаем выполнение текущей итерации и переходим к следующей
    continue;
  }
  // выведем в консоль сам символ и его индекс
  console.log(i + ' => ' + str[i]);
}
// данный цикл выведет в консоль: 2 => "о", 5 => "а", 10 => "о", 12 => "а"
```

Цикл for...of (новинка в ES6)

Цикл `for...of` появился в стандарте ES6. Предназначен он для перебора **итерируемых объектов**, т.е. объектов, в которых реализован метод `Symbol.iterator`. Этот метод ещё называют **итератором**. Именно его и использует цикл `for...of` для перебора объектов.

Метод `Symbol.iterator` имеется у `String`, `Array`, `Map`, `Set`, `arguments`, `NodeList` и других объектов.

Пример использование цикла `for...of` для посимвольного перебора строки:

```
// переменная, содержащая строку
let str = 'Новый';
// посимвольный перебор строки
for (let char of str) {
  console.log(char);
}
// в консоль будет выведено: "Н", "о", "в", "ы", "й"
```

Пример использование цикла `for...of` для перебора коллекции DOM-элементов:

```
let elements = document.querySelectorAll('p');
for (let element of elements) {
  console.log(element);
}
```

Пример использование цикла `for...of` для перебора массива:

```
// массив
let superHeroes = ['Iron Man', 'Thor', 'Hulk'];
// перебор массива
for (let value of superHeroes) {
  console.log(value);
}
// в консоль будет выведено: "Iron Man", "Thor", "Hulk"
```

Чем цикл for...of отличается от for...in

Первое отличие цикла `for...of` от `for...in` заключается в том, что он может применяться только для итерируемым объектов, т.е. объектов, в которых реализован итератор (`Symbol.iterator`). Цикл `for...in` итератор не использует. Он предназначен для перебора любых объектов.

Второе отличие заключается в том, что цикл `for...of` перебирает объект так, как это определено в итераторе. Например, в `Array` итератор реализован так, что цикл `for...of` пройдёт только по значениям в массиве и не будет включать в перебор другие (не индексные) свойства. Цикл `for...in` организован по-другому, он перебирает все перечисляемые свойства (имена ключей) объекта, в том числе и наследуемые.

Рассмотрим эти отличия. Для этого возьмём предыдущий пример и добавим к нему пользовательское свойство, например, `hero` и установим ему значение `'Wasp'`.

```
let superHeroes = ['Iron Man', 'Thor', 'Hulk'];
superHeroes.hero = 'Wasp';
```

При использовании `for...of` он переберёт все значения этого массива:

```
// цикл for...of
for (let value of superHeroes) {
  console.log(value);
}
// в консоль будет выведено: "Iron Man", "Thor", "Hulk"
```

При использовании `for...in` он переберёт все перечисляемые имена ключей этого объекта:

```
// цикл for...in
for (let key in superHeroes) {
  console.log(key);
}
// в консоль будет выведено: 0, 1, 2, "hero"
```

Чтобы получить значение ключа по его имени можно воспользоваться квадратными скобками:

```
// цикл for...in
for (let key in superHeroes) {
  console.log(superHeroes[key]);
}
// в консоль будет выведено: "Iron Man", "Thor", "Hulk", "Wasp"
```

Задачи по циклам

1. Написать с помощью цикла while «переворот» числа. Другими словами, нужно создать новое число, у которого цифры шли бы в обратном порядке (например: 472 -> 274).

2. Найти самую большую цифру в целом числе.

3. Вычислить сумму первой и последней цифр целого числа.