

ЦИКЛЫ В JAVASCRIPT




АЛЕНА БАТИЦКАЯ



АЛЕНА БАТИЦКАЯ

 [ABatitskaya](#)

 [@alenabat](#)



ПЛАН ЗАНЯТИЯ

1. Повторение – мать учения
2. Что такое циклы
3. Как работают циклы
4. Обход элементов массива с помощью цикла
5. Заполнение массива с помощью цикла



ПОВТОРЕНИЕ – МАТЬ УЧЕНИЯ

ВСПОМНИМ ПРОШЛЫЙ МАТЕРИАЛ

Что выведет консоль?

```
1 let array = ['Работает?', 'Не работает?', 'Не', 'трогай.'];  
2  
3 array.splice(1, 1);  
4  
5 console.log( array );
```

КОНСОЛЬ ВЫВЕДЕТ

```
[ 'Работает?', 'Не', 'трогай.' ]
```

Метод **splice** позволяет удалять, изменять и вставлять элементы. В рамках нашей задачи он удалит элемент по индексу 1. А так как по этому индексу находится элемент со значением **'Не работает?'**, то удалится именно он.



ЧТО ТАКОЕ ЦИКЛЫ

ЧТО ТАКОЕ ЦИКЛ?

Цикл — это инструмент, позволяющий делать однотипное действие много раз.

Пример: перебрать все числа от 1 до 10.

Примеры из жизни:

- Пока тарелка супа не опустела, мы съедаем одну ложку;
- Самолеты летают по кругу над аэропортом до тех пор, пока не разрешат посадку.

А какие примеры циклов знаете вы?

ИЗ МИРА ПРОГРАММИРОВАНИЯ

В программировании мы можем встретить вот такой реализованный пример цикла где выводится список задач:

Начать **новый** список

Enter

- × ☐ 1. Разобраться в программировании
- × ☐ 2. Погладить кота
- × ☐ 3. Выучить JavaScript
- × ☒ ~~Написать свой первый скрипт~~

КАКИЕ ЦИКЛЫ БЫВАЮТ?

В JavaScript существуют следующие операторы цикла:

- **for** — используется когда вы заранее знаете, сколько раз вам нужно что-то сделать;
- **while** — используется когда вы не знаете, сколько раз нужно что-то сделать.

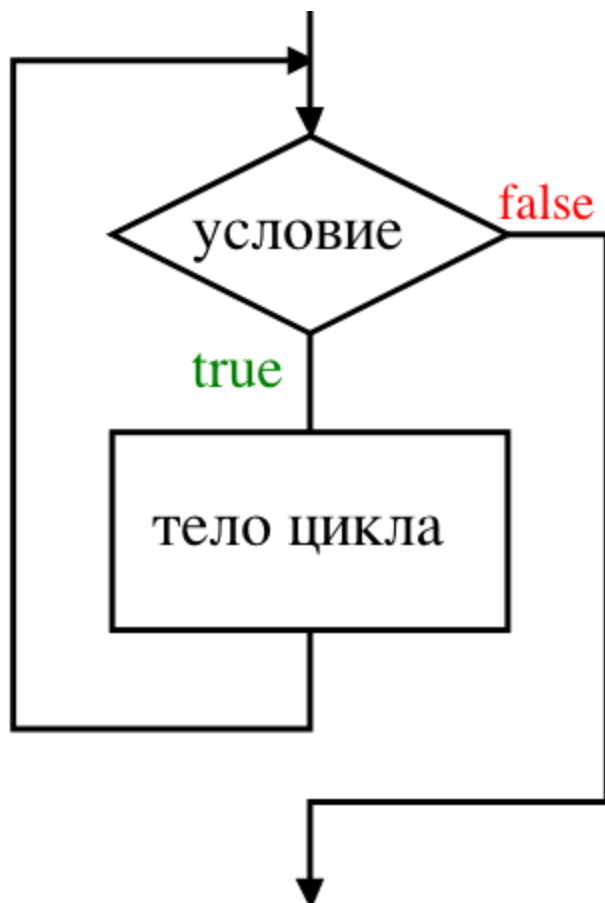
Каждый оператор цикла выбирается под разные условия поставленной задачи. Таким образом оптимально выбирается способ перебора.

Повторение цикла по-научному называется «итерация»



КАК РАБОТАЮТ ЦИКЛЫ

ПРЕДСТАВИМ ЦИКЛ ВИЗУАЛЬНО



ЦИКЛ `while...`

Это цикл с предварительной проверкой условного выражения. Проще говоря у него все важное собрано в одном месте -> пока выполняется условие срабатывает *инструкция*.

Цикл имеет такой синтаксис:

```
1  while (условие) {  
2      инструкция  
3  }
```

ЦИКЛ `while...`

Выведем также 5 чисел:

```
1  let i = 0;
2
3  while (i < 5) {
4      console.log(i);
5      i++;
6  }
```

ВАЖНО

Пока условие выполняется срабатывает инструкция в теле цикла!!!

ЧТО ЕЩЕ ЗА `i++`?

В коде мы воспользовались странной записью `i++`. Это сокращенный способ записать выражение `i=i+1`.

И то, и другое выражение увеличивает значение переменной на единицу. Используйте то, которое вам кажется удобнее.

Выражение `i++` называется инкремент. Есть еще декремент `i--`, он уменьшает значение переменной на единицу.

РАССМОТРИМ ЦИКЛ `while` ПОДРОБНЕЕ

```
1  let i = 0;  
2  while (i < 5) {  
3      console.log(i);  
4      i++;  
5  }
```

1. Инициализация **let i = 0;**
2. Проверка условия **i < 5;**
3. Завершающая операция **i++;**

КАК ВЫГЛЯДИТ ЦИКЛ `for` ?

Общий вид цикл `for` имеет такой:

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

1. Начало — на этом этапе происходит инициализация или объявление переменных. Это создание счетчика для нашего цикла.
2. Условие — выражение, выполняющееся на каждой интерации цикла. Если выражение истинно, цикл выполняется.
3. Шаг — выражение которое выполняется в самом конце цикла. Используется для обновления или увеличения переменной счётчика.

КАК ВЫГЛЯДИТ ЦИКЛ `for` ?

Итак мы поняли как выглядит цикл, так давайте выведем 5 чисел:

```
1  for (let i = 0; i < 5; i++) {  
2      console.log(i);  
3  }  
4  // 0 1 2 3 4
```

А МОЖНО ПОДРОБНЕЕ?

Рассмотрим более пристально, что делает каждое выражение:

```
1  for (let i = 0; i < 5; i++) {  
2      console.log(i);  
3  }
```

1. Инициализация **let i = 0;** — объявление переменной-счётчика.
2. Проверка условия **i < 5;** — условное выражение. В данном примере проверка условия идёт до тех пор, пока значение счётчика меньше 5.
3. Завершающая операция **i++** — операция приращения счётчика, увеличивает значение переменной **let i** на единицу. Могут использоваться и другие выражения.

КАК ПРОИСХОДИТ ПРОЦЕСС ВЫПОЛНЕНИЯ ВСЕХ УСЛОВИЙ И ВЫРАЖЕНИЙ?

Отличный вопрос, и пожалуй самый главный для 100% понимания работы циклов!

1) стартуем отсюда: начинаем перебор с 0

2) если условие верно

```
1  for (let i = 0; i < 5; i++ ) {  
2    console.log(i);  
3  }
```

4) переходим к следующему элементу

3) выполняем все, что есть в фигурных скобках

Давайте закрепим понимание цикла `for` на repl.it!

МЕНЯЕМ ШАГ

В примере мы прибавляли к текущему i единицу, чтобы перейти к следующему элементу. Для этого мы использовали выражение $i++$. В некоторых задачах нужно перемещаться с другим шагом. Давайте попробуем написать такие шаги:

- после каждой итерации i увеличивается на 2;
- после каждой итерации i удваивается;
- уменьшаем i на единицу.

ОБХОД ЭЛЕМЕНТОВ МАССИВА С ПОМОЩЬЮ ЦИКЛА

ОБХОД МАССИВА С ПОМОЩЬЮ ЦИКЛА

В основном циклы используются для выполнения **итераций по элементам массивов**.

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

Рассмотрим работу на примере массива имен.

```
1 | let names = ["Sasha", "Katya", "Vika", "Maria"];
2 | for (let i = 0; i < names.length; i++) {
3 |     console.log(names[i]);
4 | }
```

ЦИКЛЫ – ОЧЕНЬ ГИБКИЕ СТРУКТУРЫ

Например, мы можем печатать каждый второй элемент, двигаясь с шагом два:

```
1 let names = ["Sasha", "Katya", "Vika", "Maria"];
2 for (let i = 0; i < names.length; i = i+2) {
3     console.log(names[i]); // напечатаем только Сашу и Вику
4 }
```

Или по-разному выводить на печать в зависимости от индекса:

```
1 let names = ["Sasha", "Katya", "Vika", "Maria"];
2 for (let i = 0; i < names.length; i++) {
3     if (i % 2 === 0) { // для четных индексов
4         console.log("Это – " + names[i]);
5     } else {
6         console.log("Там – " + names[i]);
7     }
8 }
```




ЗАПОЛНЕНИЕ МАССИВА С ПОМОЩЬЮ ЦИКЛА

ЗАПОЛНЕНИЕ МАССИВА С ПОМОЩЬЮ ЦИКЛА

Помимо простого перебора элементов массива, мы можем выполнять и другие инструкции. Например заполнение.

Для этого создадим пустой массив, и в каждой итерации воспользуемся уже известным нам методом **push**:

(Кто знает, что делает метод **push** ?)

```
1  let array = [];  
2  for (let i=0; i<5; i++) {  
3      array.push(i);  
4  }  
5  console.log(array);
```

Получаем такой вывод:

```
// [ 0, 1, 2, 3, 4 ]
```

ЧЕМУ МЫ НАУЧИЛИСЬ?

1. Что такое циклы, как они пишутся и для чего нужны;
2. Узнали как работать с массивами и циклами;
3. Сделали еще один маленький шаг для человечества, но ОГРОМНЫЙ шаг для себя в изучении программирования.

РАДУЕМСЯ!



ДОМАШНЕЕ ЗАДАНИЕ

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаем в группе Facebook!
- Задачи можно сдавать по частям.
- Зачет по домашней работе проставляется после того, как приняты все **3 задачи**.

ГДЕ И КАК МОЖНО ПОИЗУЧАТЬ ЦИКЛЫ САМОСТОЯТЕЛЬНО

- [Codewars](#), programming challenges;
- [Циклы while, for](#), `learn.javascript.ru`;
- [Массивы: методы](#), `learn.javascript.ru`.

ЧТО ПОЧИТАТЬ

- [Документация на MDN](#);
- <https://learn.javascript.ru/while-for> — `learn.javascript.ru`.



А ПОСМОТРЕТЬ?

- <https://www.youtube.com/watch?v=ITr-SzUIDpQ> – обзор циклов от Sorax;
- <https://www.youtube.com/watch?v=orAS-MBh5f4> – хороший обзор на английском.



Спасибо за внимание! Время задавать вопросы

АЛЕНА БАТИЦКАЯ

