

Циклы и массивы

Урок 4





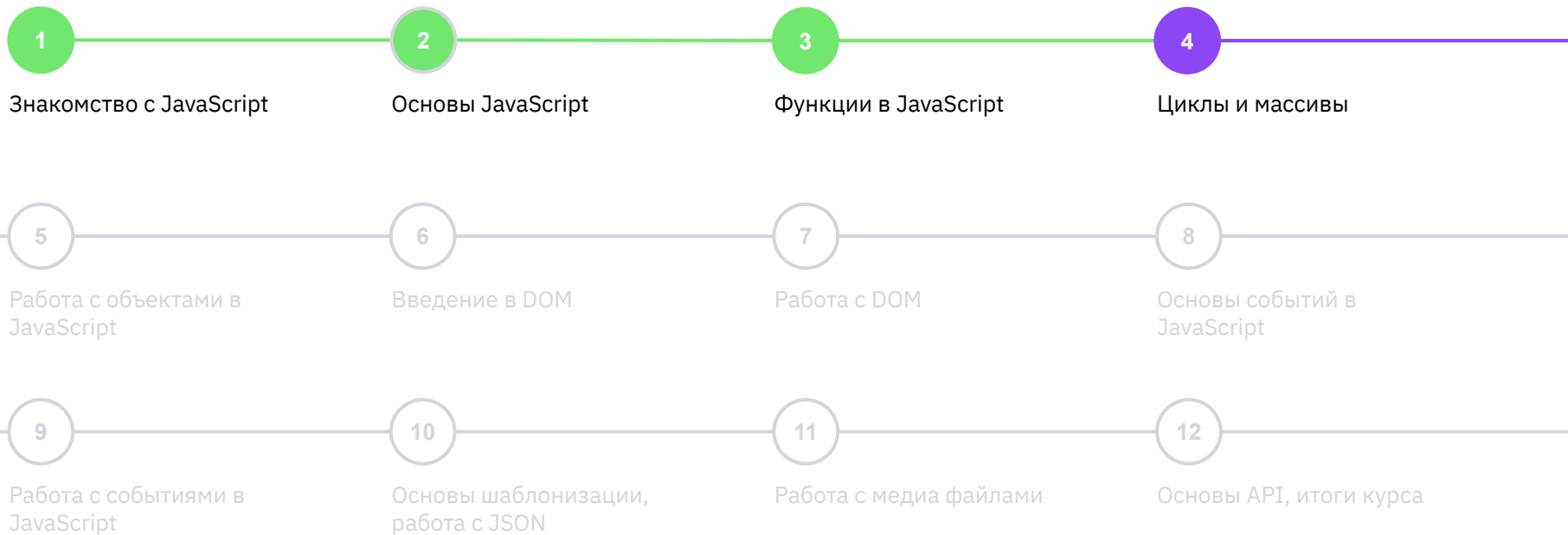
Кадочников Алексей

Frontend-разработчик

- ☀ Веб-разработчик со стажем более 9 лет
- ☀ Преподаватель GeekBrains с 2015 года
- ☀ Автор курсов по Frontend на портале Geekbrains
- ☀ Работал в таких компаниях, как VK и Wizard-C








План курса





Что будет на уроке сегодня

-  Что такое циклы
-  Виды циклов
-  Что такое массивы
-  Методы работы с массивами
-  Комбинирование циклов и массивов



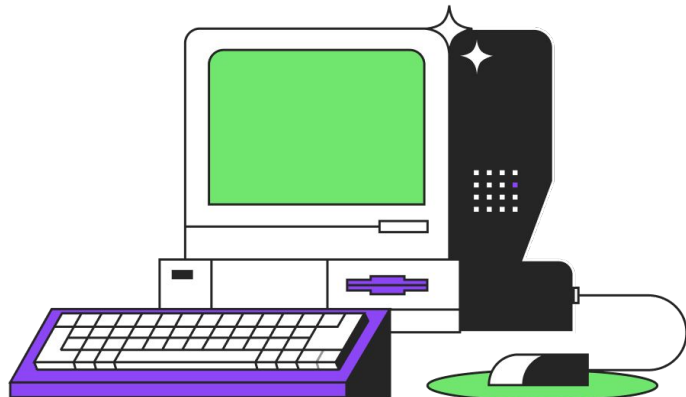
Циклы





Что такое цикл

Цикл – разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций. Говоря простым языком, цикл – это конструкция, которая заставляет определённую группу действий повторяться до наступления нужного условия.





Цикл состоит из следующих шагов

1. Инициализация переменных цикла в начале цикла.
2. Проверка условия выхода на каждой итерации (до или после неё).
3. Исполнение тела цикла на каждой итерации.
4. Обновление счётчика итераций.



Циклы в JavaScript



Циклы в JavaScript

Как и у любого языка программирования, в JavaScript также есть циклы. Давайте рассмотрим несколько видов циклов на интересных примерах. Одно из обычных применений циклов — обход элементов массива.





Цикл while

Цикл `while` является примером цикла с предусловием. Это цикл, который выполняется, пока истинно некоторое условие, указанное перед его началом. Поскольку условие проверяется до выполнения самой первой итерации, вполне может не выполниться ни одной итерации, если условие изначально ложно.

```
1 while (condition) {  
2 //Тело цикла  
3 }  
4
```



Цикл while

Тело цикла, содержащее операторы, будет выполняться до тех пор, пока истинно условие, указанное в начале цикла. Алгоритм такого цикла в сравнении с базовым будет выглядеть так:

1. Проверка условия.
2. Выполнение тела, если условие истинно. Выход, если условие ложно.



Цикл while

Для управления циклом обычно требуется одна или несколько переменных. К примеру, некое значение `boolean`, которое обращается в `false` при достижении некоего граничного условия. Давайте рассмотрим пример: наша задача в том, чтобы программа выводила в консоль значения начиная с единицы до значения `N`, которое как раз пользователь введет с клавиатуры.

```
1 const number = Number(prompt('Введите значение N'));
2 let i = 1;
3 while (i ≤ number) {
4     console.log(i++);
5 }
6
```



Цикл `do..while`

Цикл `do...while` – это уже цикл с постусловием, работающий по алгоритму:

1. Выполнение блока операторов.
2. Проверка условия.
3. Выход, если условие ложно.



Цикл do..while

Цикл с постусловием — это цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз.

Закономерный вопрос: зачем нам еще один вид цикла, если он так сильно похож на while? Давайте для этого рассмотрим пример:

```
1 let pass = Number(prompt('Введите пароль в числовом формате'));
2 while (pass === 123) {
3     pass = Number(prompt('Введите пароль в числовом формате'));
4 }
5
```



Цикл do..while

Пример такой же программы, только с помощью do while

```
1 let pass;  
2 do {  
3     pass = Number(prompt('Введите пароль в числовом формате'));  
4 } while (pass !== 123);
```



Цикл for

Цикл for – это цикл со счётчиком. Сам по себе этот цикл представляет прекрасный пример лаконичной организации кода, имея максимально схожий вид в большинстве языков программирования.

Инициализация, проверка и обновление — это три ключевых операции, выполняемых со счетчиком. Цикл for сводит эти три шага воедино:

```
1 for(инициализация; проверка; инкремент)
2 {
3     инструкция
4 }
5
```




Цикл for

Если провести аналогию с известным нам уже циклом **while**, получится вот такой псевдокод:

```
1 инициализация;  
2 while(проверка)  
3 {  
4     инструкция;  
5     инкремент;  
6 }  
7
```



Цикл for

Итак, в цикле for перед началом цикла выполняется инициализация, в которой обычно и объявляется переменная-счётчик. Проверка вычисляется в начале каждой итерации цикла. Если она возвращает true, цикл продолжается, в противном случае – останавливается. По окончании итерации производится инкрементирование счетчика.

И снова в цикле выведем числа от 1 до N, но уже с применением цикла for:

```
1 const number = Number(prompt('Введите значение N'));
2 for (let i = 1; i ≤ number; i++) {
3   console.log(i);
4 }
5
```



Цикл do..while

Давайте подведём итоги работы с циклами, какой же выбрать и какой является самым популярным, ответ тут простой, если говорить про популярность, то это определённо цикл for, так что тут вне конкуренции. Если же полагаться на рациональный выбор, то давайте рассмотрим инструкцию:

1. Если нас интересует бесконечный цикл

```
1 while (true) {  
2 //будет выполняться бесконечное количество раз  
3 }  
4
```

2. Если нам требуется сначала что-то сделать, а потом уже реализовать проверку, то тут к нам на помощь придёт do while
3. Если нас интересует счетчик или возможно потребуется считать что либо, то это определённо for

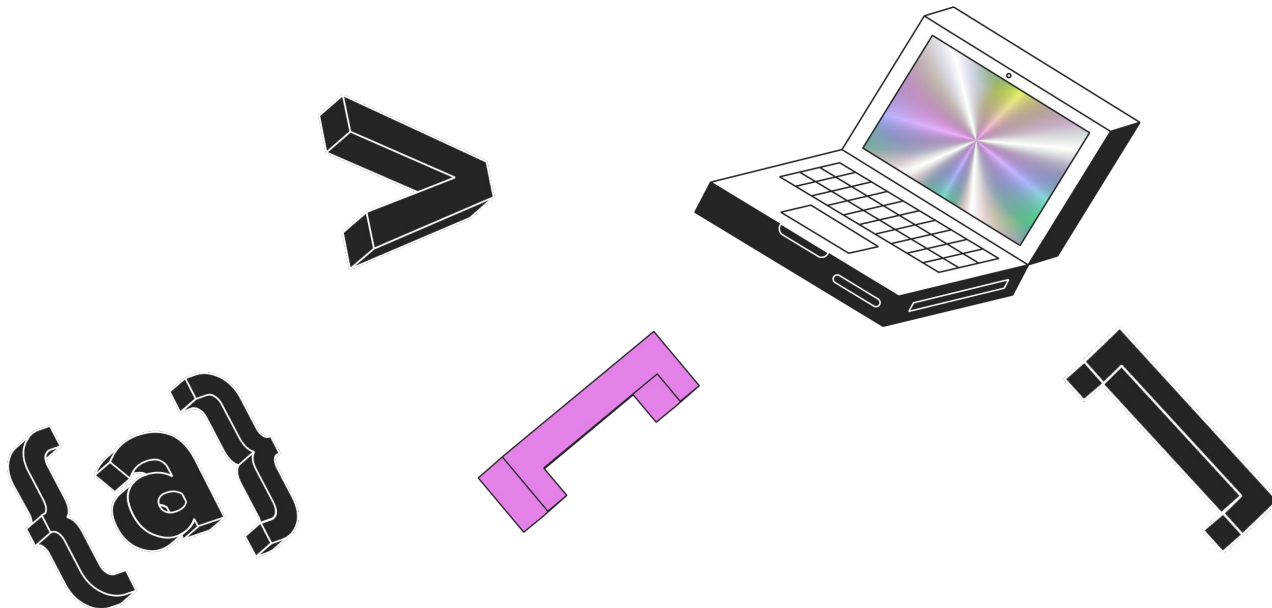


Массив и его методы



Массив и его методы

Массив — это упорядоченный список элементов. Массивы дают удобство при обработке однородных данных, позволяя производить операции в циклах для преобразования, фильтрации или сбора нужной информации из исходного массива.





push

Метод **push** принимает один или несколько аргументов, которые будут добавлены в конец массива.

Давайте добавим в наш массив студентов трёх студентов: Иванова, Петрова и Сидорова:

```
1 const students = [];  
2  
3 students.push('Иванов');  
4 students.push('Петров');  
5 students.push('Сидоров');  
6  
7 console.log(students ); // ['Иванов', 'Петров', 'Сидоров']  
8
```



push

Еще один пример, когда мы добавляем сразу несколько элементов:

```
1 const students = [];  
2  
3 students.push('Иванов', 'Петров', 'Сидоров');  
4  
5 console.log(students ); // ['Иванов', 'Петров', 'Сидоров']  
6  
7
```



pop

Метод **pop** позволяет извлечь из массива последний элемент, при этом он удаляется из массива, а если массив пустой, то вернется undefined. Этот метод часто используется чтобы получить последний элемент массива,

```
1 const students = ['Иванов', 'Петров', 'Сидоров'];  
2  
3 const lastStudent = students.pop();  
4  
5 console.log(lastStudent); // 'Сидоров'  
6 console.log(students); // ['Иванов', 'Петров']  
7  
8
```




pop

Также этот метод позволяет получать последний элемент из строки, которую можно разделить на части, например у нас есть полный путь до файла **C:/projects/bestProject/src/images/background-image.png** и мы хотим из этого пути получить только файл. Всю строку можно разбить на части, разделенные символом косой черты (слеш “/”), и взять последнюю такую часть, давайте посмотрим пример кода:

```
1 const filePath = "C:/projects/bestProject/src/images/background-  
  image.png";  
2 const fileName = filePath.split('/').pop(); // Разделим строку на  
  составляющие и превратим её в массив по средствам split('/'), а  
  потом уже вызовем новый метод pop()  
3  
4 console.log(fileName); // "background-image.png"  
5
```



shift

Этот метод произошел от английского слова shift — сдвигать. Данный метод извлекает нулевой элемент из массива, при этом сдвигает все оставшиеся элементы массива на одну ячейку влево. Получим первого студента, записавшегося на курс.

```
1 const students = ['Иванов', 'Петров', 'Сидоров'];  
2  
3 const firstStudent = students.shift();  
4  
5 console.log(firstStudent); // 'Иванов'  
6 console.log(students); // ['Петров', 'Сидоров']  
7
```



shift

Также методом **shift** можно получить имя диска, из полного пути до файла:

```
1 const filePath = "C:/projects/bestProject/src/images/background-  
  image.png";  
2 const diskName = filePath.split('/').shift(); // Разделим строку на  
  составляющие и превратим её в массив по средствам split('/'), а  
  потом уже вызовим новый метод shift()  
3  
4 console.log(diskName); // "C:"  
5
```



slice

На практике часто бывает необходимо сделать копию массива, но т.к. массив — это объект, поэтому переменная массива — это указатель на область памяти, и мы не можем просто присвоить его новой переменной. У нас тогда будет две переменных, ссылающихся на один и тот же массив, что может привести к изменению исходного массива, когда мы будем работать с новым. Давайте посмотрим на примере:

```
1 const students = ['Иванов', 'Петров', 'Сидоров'];
2 // Попробуем скопировать массив students в новую переменную.
3 const students2 = students;
4 // Добавим в новую переменную нового студента.
5 students2.push('Белкин');
6
7 console.log(students); // ['Иванов', 'Петров', 'Сидоров', 'Белкин']
8 console.log(students2); // ['Иванов', 'Петров', 'Сидоров', 'Белкин']
9
```



slice

Как видно из результатов, мы меняли второй массив, но вместе с ним изменился и первый, т.к. вторая переменная получила тот же указатель на данные, что и первая. Чтобы скопировать массив целиком, мы можем воспользоваться методом **slice**, вызванным без аргументов, тогда он вернет нам полную копию массива:

```
1 const students = ['Иванов', 'Петров', 'Сидоров'];
2 // Попробуем скопировать массив students в новую переменную.
3 const students2 = students.slice();
4 // Добавим в новую переменную нового студента.
5 students2.push('Белкин');
6
7 console.log(students); // ['Иванов', 'Петров', 'Сидоров']
8 console.log(students2); // ['Иванов', 'Петров', 'Сидоров', 'Белкин']
9
```



slice

Помимо копирования массивов, метод **slice** (с английского — отрезать) позволяет отрезать от исходного массива часть, при этом исходный массив остается целым, а метод возвращает новый массив. Например когда нам нужно получить 2-х первых студентов группы, мы можем сделать так:

```
1 const students = ['Иванов', 'Петров', 'Сидоров', 'Белкин'];  
2 const firstTwoStudents = students.slice(0, 2);  
3  
4 console.log(firstTwoStudents ); // ['Иванов', 'Петров']  
5
```



indexOf

Метод **indexOf** вычисляет индекс определенного элемента, совпадающего со значением, переданным в качестве аргумента. Проще говоря, выполняет поиск элемента. В случае успеха метод возвращает индекс первого найденного элемента, в противном случае возвращается значение -1.

Давайте попробуем проверить, есть ли в нашей группе студентов, студент по фамилии Ivanov:

```
1 const students = ["Ivanov", "Petrov", "Sidorov", "Alexandrov",  
2   "Belkin", "Avdeev"];  
3 // Неправильная проверка.  
4 if (students.indexOf("Ivanov")) {  
5   console.log("Среди студентов есть Иванов!"); // Ничего не будет  
6   // выведено, т.к. Иванов является нулевым элементом массива, а ноль  
7   // приводится к false значению, поэтому такая проверка не сработает.  
8 }  
9 // Правильная проверка.  
10 if (students.indexOf("Ivanov") !== -1) {  
11   console.log("Среди студентов есть Иванов!"); // "Среди студентов  
12   // есть Иванов!"  
13 }  
14 const indexOfBelkin = students.indexOf("Belkin"); // 4  
15
```



Разминка

Задание Дана переменная $a = 4$, ее необходимо добавить в конец массива $[1, 2, 3]$



Разминка

Задание Дана переменная $b = 0$, ее необходимо добавить в начало массива $[1, 2, 3]$

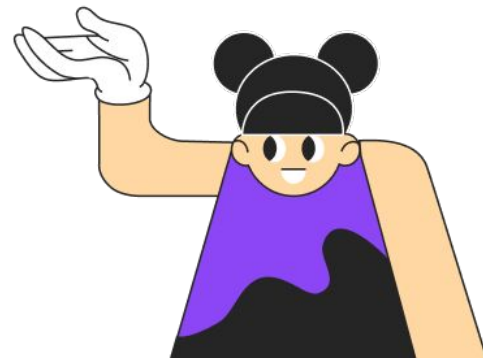


Комбинирование массивов и циклов



Комбинирование массивов и циклов

Вам необходимо запомнить одно простое правило:
работа с массивами очень часто связана с циклами,
поэтому мы очень часто будем использовать циклы, для
перебора элементов массива





Комбинирование массивов и циклов

Вам необходимо запомнить одно простое правило: работа с массивами очень часто связана с циклами, поэтому мы очень часто будем использовать циклы, для перебора элементов массива

Теперь давайте рассмотрим простую задачу, которая, однако, уже будет максимально приближена к программированию, которому мы привыкли видеть. У нас будет массив названий товаров, как в обычном интернет-магазине, и при клике на кнопку мы будем удалять последний из товаров.

```
1 const products = ['Кофта', 'Куртка', 'Футболка', 'Брюки'];
2 for (let i = 0; i < products.length; i++) {
3     console.log(products[i]);
4 }
5
```



Комбинирование массивов и циклов

Давайте усложним задачу и сделаем кнопку, при нажатии на которую мы будем удалять последний элемент массива:

```
1 <button onclick="delProduct()">Удалить товар</button>
2
3 <script>
4   const products = ['Кофта', 'Куртка', 'Футболка', 'Брюки'];
5   console.log('Список всех товаров');
6   for (let i = 0; i < products.length; i++) {
7     console.log(products[i]);
8   }
9
10  function delProduct() {
11    products.pop(); // удаляем последний элемент массива
12    console.log('Список товаров после нажатия на кнопку');
13    for (let i = 0; i < products.length; i++) { // выводим
      обновленный список товаров
14      console.log(products[i]);
15    }
16  }
17 </script>
18
```

Создаём функцию удаления товара, которая используем метод **pop()** и выводим обновленный список товаров с помощью цикла for



Разминка

Задание Дан массив $[1, 2, 3]$ необходимо найти сумму элементов массива









Разминка

Задание Дан массив [1, 2, 3] необходимо найти элементы массива большие и равные 3



Итоги урока

-  Рассмотрели, что такое циклы
-  Узнали, какие виды циклов бывают
-  Рассмотрели, что такое массивы
-  Разобрали методы работы с массивами
-  Рассмотрели комбинирование циклов и массивов

Спасибо 
за внимание

