

Циклы в JavaScript



JavaScript
Courses

courses.dp.ua

```

2 //while - цикл с проверкой условия на входе;
3 while(a > b){
4     //.....
5 }
6
7
8 //do-while - цикл с проверкой условия на выходе;
9 do{
10     //.....
11 }while(a != b);
12
13 //for - цикл со счётчиком;
14 for(var i = 0; i < 10; i++){
15     //.....
16 }
17
18 //for-of - цикл перебора значений массивов и псевдомассивов;
19 let arr = [10, 35, 70, 90, 120];
20 for(let value of arr){
21     //.....
22 }
23
24 //for-in - цикл перебора ключей объекта.
25 let ob = { name: "Jhon", lastName: "Smith", age: 28, city: "Dnipro" };
26 for(let key in ob){
27     //.....
28 }
29

```

Циклы в JavaScript

JavaScript содержит большой набор из 5 циклов (в классическом понимании цикла как средства повторения фрагмента кода) и десятков «цикло-подобных» конструкций. Циклы в JavaScript ориентированны на широкий спектр задач: циклы по условию (на входе и на выходе), цикл со счётчиком, циклы для перебора ключей и значений в структурах данных.

Цикл do-while

```
2
3   let secret = "7833";
4   let code;
5
6   do{
7
8       code = prompt("Enter code:");
9
10      if(code == secret){
11          alert("Access is allowed.")
12      }else{
13          alert("Access denied!");
14      }
15
16  }while(code != secret);
17
```

Цикл **do-while** – универсальный цикл с условием, цикл выполняются (повторные итерации цикла выполняются) пока условие будет истинным (**true**). В цикле **do-while** условие проверяется на при завершении каждой итерации цикла (на выходе), такая конструкция цикла обеспечивает, что **тело цикла do-while выполнится минимум один раз.**

Подробнее: <https://learn.javascript.ru/while-for>

Цикл `while`

```
2
3   let balance = 1000;
4   let payment_sum = 29.99;
5
6   while(balance >= payment_sum){
7
8       //...some payment...
9       balance -= payment_sum;
10      console.log("Payment OK, balance:", balance);
11
12  }
13
```

Цикл **while** – универсальный цикл с условием, цикл выполняется (повторные итерации цикла выполняются) пока условие будет истинным (**true**). В цикле **while** условие проверяется перед началом каждой итерации цикла при такой конструкции может возникнуть ситуация при которой тело цикла ни разу не выполнится (если проверка условия перед первым шагом сразу дать **false**).

Подробнее: <https://learn.javascript.ru/while-for>

Цикл for

```
2
3   let users = ['Jane', 'Amanda', 'Maria', 'Bill', 'Jhon'];
4
5  ✓   for(let i = 0; i < users.length; i++){
6      |       console.log(`User #${i}: ${users[i]}`);
7      |
8      }
```

Цикл **for** – универсальный цикл с условием, также называемый циклом со счётчиком. выполняются (повторные итерации цикла выполняются) пока условие будет истинным (**true**). В цикле **for** условие проверяется перед началом каждой итерации цикла при такой конструкции может возникнуть ситуация при которой тело цикла ни разу не выполнится (если проверка условия перед первым шагом сразу дать **false**). В цикле предусмотрен удобный механизм ведения счётчика итерацией цикла. Цикл **for** традиционно применялся для перебора массивов и псевдомассивов (до появления цикла **for-of**)

Подробнее: <https://learn.javascript.ru/while-for>

```
2
3 let users    = ['Jane', 'Amanda', 'Maria', 'Bill', 'Jhon'];
4
5 let set      = new Set();
6
7 set.add("Samsung");
8 set.add("Apple");
9 set.add("Xiaomi");
10
11 let str      = "Hello world!";
12
13 for(let item of users){
14 |   console.log("Array element:", item);
15 | }
16
17 for(let item of set){
18 |   console.log("Set element:", item);
19 | }
20
21 for(let item of str){
22 |   console.log("String element:", item);
23 | }
24
```

Цикл **for-of** (ES2015)

Цикл **for-of** – предназначен для перебора значений итерируемых (*iterable*) структур данных (Массивов, Set, Map и псевдомассивов, **но не объектов**). Цикл **for-of** – берёт на себя нумерацию и контроль элементов. Но не даёт возможность поменять элемент в массиве.

Подробнее:

<http://jsraccoon.ru/es6-for-of-loop>

Операторы break / continue

Операторы break и continue

```
2
3  ✓ while(a > b){
4      //...
5  ✓   if(a == 100){
6      continue;
7      }
8
9  ✓   if(b == 100){
10      break;
11      }
12   }
13
```

Оператор **break** позволяет прервать цикл, оператор **continue** позволяет завершить текущий шаг (итерацию) цикла и перейти к следующей. Могут применяться во всех 5-ти видах циклов.

Подробнее: <https://learn.javascript.ru/while-for#metki-dlya-break-continue>

Немного практики #1

«Заморозки»

```
let winter = [-48, -46, 48, 27, -20, -35, 43, 4, 9, 10, 41, -46, -4, 0, -38, -49, 25,  
-46, -48, -23, -25, -22, 12, 38, 19, -20, 26, 4, 19, 23, 26, -41, 4, -13, -9, -11, -7,  
38, 27, 41, 14, -35, -38, -44, -44, -22, -24, 29, -32, 41, 7, -25, 3, 27, -45, 10, 48,  
8, -34, -49, 17, -16, 41, -11, -50, -6, -34, 20, 14, -18, 39, -28, -33, -27, -48, 40,  
-37, -44, 0, 46, 36, -34, -50, 8, -3, 26, 40, 10, -36, 24];
```

Есть массив с данными о средних температурах за 90 дней зимы. Программа должна определить какая была продолжительность максимального периода заморозков.

Заготовка лежит в репозитории этого занятия в каталоге: </example/ex01.html>

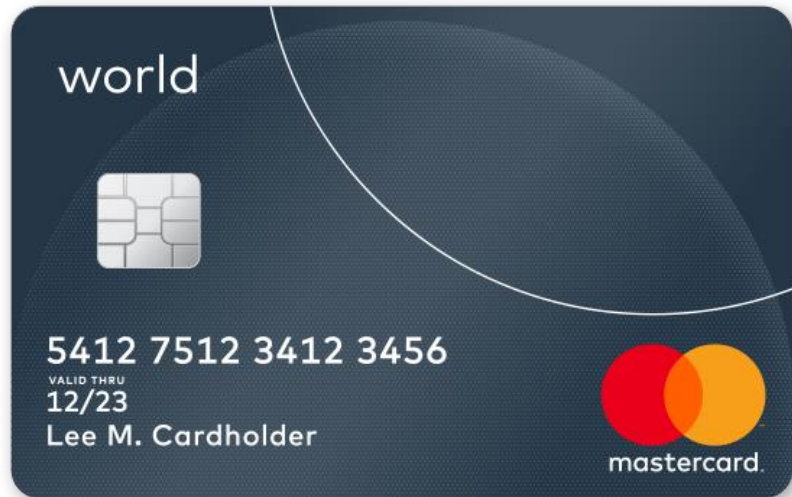
Немного практики #2

Кредитный калькулятор v.1

Заданы: Есть сумма кредита, годовая процентная ставка, и срок кредитования в месяцах. **Рассчитать** ежемесячные платежи (сколько в каждом месяце будет платить заёмщик, указав сколько из суммы ежемесячного платежа идёт на погашение тела кредита, а сколько на погашение процентов, а также, сколько остаётся долга по телу кредита) по **классической** схеме.

Немного практики #3

Алгоритм Луна



VISA 4916 5526 5398 1949








5357 6872 3409 1447

Алгоритм Луна проверяет контрольную сумму числа, широко применяется для проверки корректности номера банковских карт.

Задача: пользователь вводит номер банковской карты, необходимо проверить не ошибся ли он.

Подробнее: https://uk.wikipedia.org/wiki/Алгоритм_Луна

Генератор номера карты

 Visa	 MasterCard	 Discover	 AmericanExpress	 JCB
✓ 4412530595659632	✓ 5287324989755118	✓ 6011139619422678	✓ 340849911182813	✓ 3539584124038594
✓ 4813431262431071	✓ 5369658110635785	✓ 6011117040432748	✓ 345673843441369	✓ 3588422734539547
✓ 4381493988886337	✓ 5153000135610537	✓ 6011406220044898	✓ 345616475358716	✓ 3538044621974255
✓ 4739306813042299	✓ 5327520507510974	✓ 6011774117039986	✓ 375103335418603	✓ 3528852467705472
✓ 4464941706819170	✓ 5155034861872910	✓ 6011069037122495	✓ 375423401400255	✓ 3579534744222947
Generate Visa ➤	Generate MasterCard ➤	Generate Discover ➤	Generate AmEx ➤	Generate JCB ➤

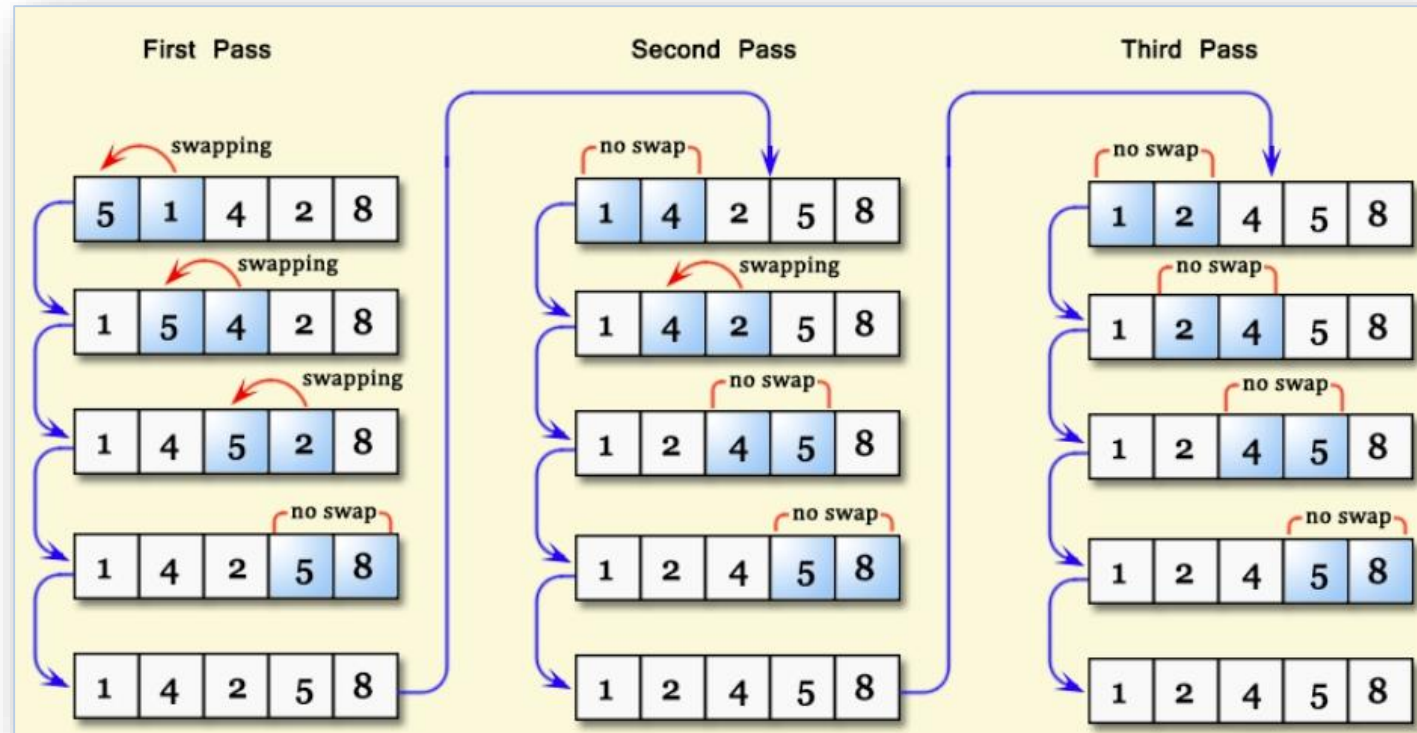
В помощь: генератор номеров банковских карт:

<https://www.getcreditcardinfo.com/>

Немного практики #4

Сортировка данных (массивов)

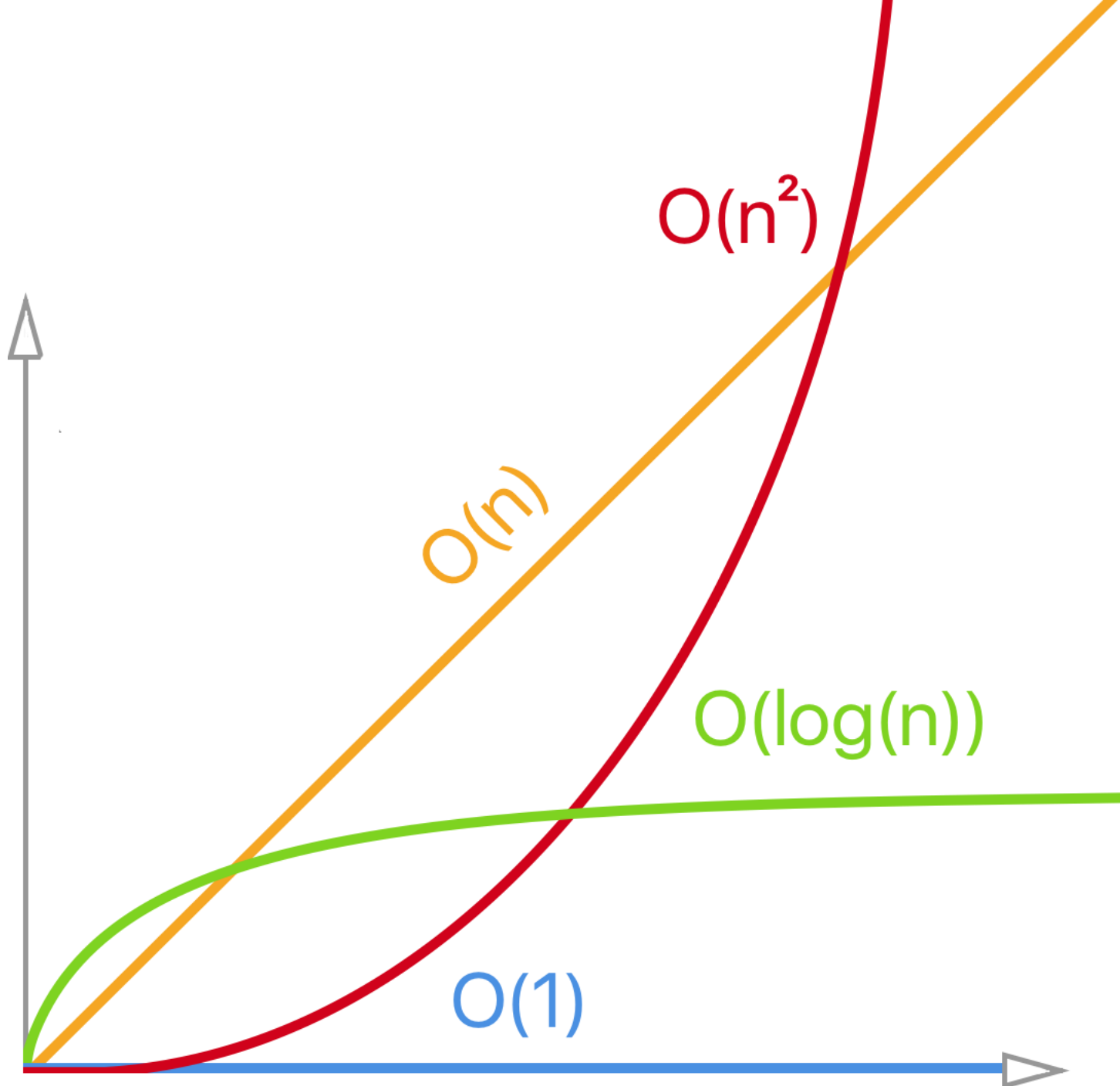
Когда необходимо внести изменения в существующий набор данных.



Классический алгоритм «пузырьковой» сортировки.

Подробнее о алгоритмах сортировки: <https://habrahabr.ru/post/204600/>

Сложность алгоритма



Оценка сложности алгоритма (концепция Big O)

Зависимость времени
выполнения (а по сути
количества операций) от
количества
обрабатываемых данных

Подробнее:

<https://habr.com/ru/post/444594/>
<https://www.youtube.com/watch?v=ZRdOb4yR0kk>

Замеры времени выполнения кода

`performance.now()`

```
2
3   let t1 = performance.now();
4
5   for(let i = 0; i < 1000000; i++){
6       //...do something HARD
7   }
8
9   let t2 = performance.now();
10
11  console.log('Time for HARD work (ms):', t2 - t1);
12
```

Метод **performance.now()** возвращает в миллисекундах временную метку. При сравнении двух и более временных меток можно определить время прошедшее между их получением.

Домашнее задание
/узнать

Современный учебник Javascript

Перед вами учебник по JavaScript, начиная с основ, включающий в себя много тонкостей и фишек JavaScript/DOM.



смотреть на Github

Поделиться:



НАЙТИ

Содержание

Первые две части посвящены JavaScript и его использованию в браузере. Затем идут дополнительные циклы статей на разные темы.

Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **пройдёте разделы первой части 2.14-2.16, 5.5, 6.1-6.3, 6.8, 6.11, 10.1**

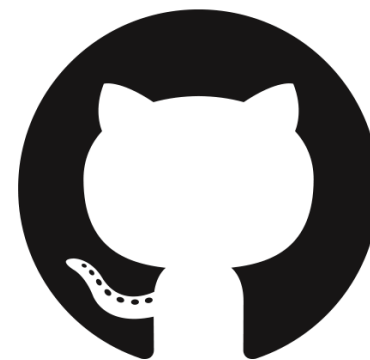
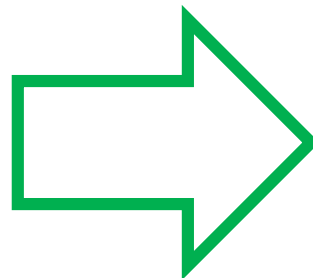
<http://learn.javascript.ru/>

Домашнее задание
/сделать

Каждое домашнее задание оформляйте в виде отдельного репозитория на GitHub, в названии которого должен быть указан код задания (например: B3).

НЕ ЗАБЫВАЕМ ОБ
ОТСТУПАХ В КОДЕ!!!

Ссылку на репозиторий сбрасывайте
в Slack в канал **#homeworks**



Если есть проблемы, вопросы, трудности, делаем тоже самое – код с проблемой заливаем на **GitHub** и ссылку на него, с описанием вопроса в **Slack**, но в канал **#trouble**

Домашнее задание #B.1

```
[{"title": "Apple", "price": "219 $"},  
{"title": "Samsung", "price": "28 $"},  
{"title": "Nokia", "price": "976 $"},  
{"title": "Meizu", "price": "742 $"},  
{"title": "Xiaomi", "price": "75 $"},  
{"title": "Huawei", "price": "542 $"},  
{"title": "Sony", "price": "901 $"},  
{"title": "LG", "price": "15 $"},  
{"title": "ASUS", "price": "383 $"}]
```

Заготовка лежит в репозитории этого занятия в каталоге: /homework/B1_template.html

*Задача: отсортировать массив со списком товаров
по возрастанию цены.*

Домашнее задание #В.2

Создайте программу проверяющую знания (умение) таблицы умножения двузначных чисел. Скрипт должен задать пользователю 12 задач на умножение **двузначных** чисел. По результатам проверки, пользователю выставляется оценка (по 12 бальной шкале), а также выводиться два списка: верных ответов, и ошибочных ответов, с указанием какой ответ был правильный.

Убедитесь, что программа генерирует именно двузначные числа для вопросов (от 10 до 99 включительно).

Домашнее задание #В.3

Кредитный калькулятор v.2

Заданы: сумма кредита, годовая процентная ставка, и срок кредитования в месяцах.

Рассчитать ежемесячные платежи (сколько в каждом месяце будет платить заёмщик, указав сколько из суммы ежемесячного платежа идёт на погашение тела кредита, а сколько на погашение процентов) по **аннуитетной** схеме.