# 

[**Общие сведения**](#_nmperkwx0l8r) **1**

[Литература](#_j2jw7cdksvnz) 1

[Требования](#_t3d1r9r8ukjl) 2

[Реализация алгоритма](#_3y9c6hkgdyn9) 2

[Имена переменных/функций и другие наименования](#_w33klwy7p2g) 2

[Дополнительные требования к коду](#_rb19xsrl3dh3) 2

[Терминология](#_yu9lxm9ad7ig) 2

[**Индивидуальные лабораторные работы**](#_iulq0s1ydcxv) **3**

[Основы основ](#_6uborcyuf4g2) 3

[Углубленный javascript](#_s8mxinmuks45) 3

[Алгоритмы и структуры данных](#_6y65z1bukbt6) 4

[Объекты](#_93173a1e8va4) 4

[Регулярные выражения](#_1adyfajh5nt2) 4

[Document Object Model](#_wowkxr2c2gtn) 5

[TypeScript](#_9619wl4ngc3a) 5

[Приобретенные скилы](#_eusgoj8q8si5) 5

[**Проектирование**](#_ol9qzzrvgz7h) **5**

[Приобретенные скилы](#_oie7wq21dy21) 5

[**Групповые лабораторные работы**](#_2zbu7f53ak57) **6**

[Single page application](#_ez67pancgzw0) 6

[Список макетов](#_xb4znh4n9pjw) 6

# Общие сведения

## Литература

1. JavaScript: Полное руководство, 7-е издание - Дэвид Флэнаган
2. Robert Martin: Clean code
3. Кнут Дональд Эрвин: Искусство программирования
4. <https://developer.mozilla.org/ru/docs/Web/JavaScript>
5. <https://learn.javascript.ru/>

Дополнительно к прочтению:

1. <https://google.github.io/styleguide/jsguide.html>
2. <https://developer.mozilla.org/en-US/docs/MDN/Guidelines/Code_guidelines/JavaScript>

## Требования

### Реализация алгоритма

Реализовывать без стандартных функций, под этим подразумевается, чтобы задача не выполнялась, заменой:

function powl(*num*, *n*){

return Math.pow(num, n);

}

Чтобы задача не решалась вызовом стандартной аналогичной функцией.

### Имена переменных/функций и другие наименования

1. Говорить зачем это нужно в коде
2. Отвечать на вопрос что храниться и для чего используется
3. Не писать имена которые не говорят ничего, например: arr, a, b, b2, b3, c, my, и другие.
4. Исключением могут быть переменные с одной буквы i, j - которые являются бегунками в цикле, и порядок их должен быть исключительно таким же. Переменная с именем “e” допускается только в одном случае при обработке исключений.
5. Всегда использовать camelCase

### Дополнительные требования к коду

1. Все фразы в языковых конструкциях должны заканчиваться точкой запятой (Пустой оператор).
2. Код должен быть написан в режиме “use strict”
3. Функция всегда должна выполнять только одно действие для чего она предназначена и возвращать результат своего выполнения. Любые побочные эффекты в функции запрещены

## Терминология

**Простое число:** Натуральное число, которое делиться нацело только на само себя и единицу. Например 3, 5, 7 и т.д.

**Числа фибоначчи:** 0 1 1 2 3 5 и т.д., каждое последующее число это сумма двух предыдущих чисел, всегда начинается с 0 и 1.

**Анаграмма:** Это разные по смыслу слова, у которых одинаковый набор букв. Например банка-кабан, мышка-камыш

**Побочные эффекты:** Это использование такого функционала который невозможно протестировать, например: ввод/вывод куда либо и другой функционал

**Процедура:** Это функция которая не имеет возвращаемого значения

# Индивидуальные лабораторные работы

### Основы основ

1. Написать функцию которая проверяет являются две строки анаграммой или нет
2. Нарисовать блок схему алгоритма анаграммы
3. Написать функцию которая вычисляет подсчет количество каждой цифре в числе
4. Написать функцию которая вычисляет подсчет уникальных слов в предложении
5. Написать функцию которая вычисляет вхождение каждого слова в предложение
6. Написать функцию которая возвращает N первых чисел фибоначчи
7. Вычислить периметр и площадь для прямоугольника, треугольника и круга
8. Вычислить факториал числа
9. Посчитать сумму всех элементов массива, только тех которые (Кратные двум, кратные трем, которые только положительные и нечетные)
10. Посчитать количество элементов массива которые (Нулевые, отрицательные, положительные, простые числа)
11. Написать функции которые преобразовывают число из десятичной системы счисления в двоичную и в обратную сторону. (Достаточно написать для целых положительных чисел)
12. Пункты 9 и 10 выполнить для двумерных массивов
13. Посчитать сумму значений чисел от min до max (всех, только тех которые кратны 3, только положительные)
14. Найти среднее значение всех элементов одномерного/двумерного массива (Среднее только тех которые четные и которые не четные)
15. Транспонировать матрицу
16. Сложить две матрицы
17. Удалить из двумерного массива строку в которой присутствует хотя бы один нулевой элемент. Для столбца аналогично реализовать.
18. Для всех заданий где можно использовать рекурсию, реализовать рекурсивные функции.
19. Написать мемоизированные функции высшего порядка, для ранее написанных своих рекурсивных функций из текущего раздела.

### Углубленный javascript

1. Написать свою реализацию функций bind, call. Новая реализация должна по функционалу работать аналогично как и соответствующие стандартные функции. Без использования стандартных функций.
2. Написать свою реализацию функций для работы с массивами, которые являются аналогами следующих функций: map, filter, reduce, forEach. Без использования стандартных функций.
3. Создать итерируемый объект, который на каждой итерации возвращает следующее значение числа фибоначчи (Реализовать с помощью итератора и генератора)

### Алгоритмы и структуры данных

1. Написать свою реализацию бинарного дерева поиска. (Возможности структуры данных должны быть: Добавить новый элемент, удалить элемент, найти элемент по его значению)
2. Написать сортировку двумя различными методами (Можно выбрать любые методы сортировки, самые простые: пузырьковая, выбором)

### Объекты

1. Ресторан, у ресторана есть сотрудники, каждый сотрудник принадлежит своему отделу. У сотрудника - имя, фамилия, номер отдела, должность, зарплата, уволенный или нет. У отдела - номер отдела, наименование
2. Посчитать сумму всех зарплат по каждому отделу
3. Посчитать среднюю зарплату по отделу
4. Найти самую большую и самую маленькую зарплату в разрезе каждого отдела и должности
5. Посчитать количество уволенных сотрудников
6. Найти отделы в которых нет руководителя (Руководитель отдела - должность) т.е. в отделе есть хотя бы один сотрудник с такой должностью
7. Клиенты банка, имеют такие характеристики - фио, активный или нет, дата регистрации в банке, счета. Существует два типа счетов: дебетовый и кредитовый. Дебитовый счет имеет текущий баланс либо он положителен либо нулевой. Кредитовый счет имеет два баланса: личные средства, кредитные средства и кредитный лимит. У каждого счета есть активность, дата активности когда заканчивается срок годности пластиковой карты. У каждого счета есть тип валюты, UAH, RUB, USD, GBP, EUR и другие. Подсчитать общее количество денег внутри банка в долларовом эквиваленте учитывая кредитные лимиты и снятие средств. Посчитать сколько всего денег в долларовом эквиваленте все клиенты должны банку. Посчитать сколько неактивных клиентов должны погасить кредит банку и на какую общую сумму. Аналогично для активных. Для получения актуальных курсов валют использовать API (которое будет предоставлено). Промисы использовать для работы с API в целях отправки запросов на сервер. Создать отдельный git-репозиторий для этого проекта и дальше работать с этим проектом в этом репозитории.

### Регулярные выражения

1. Написать регулярное выражение проверки номера телефона по формату +сс(mmm)xxx-xx-xx, Где cc - код страны, mmm - код мобильного оператора, x - номер телефона
2. Написать регулярное выражение проверки на email
3. Написать регулярное выражение проверку на сайт: <http://test.dev>
4. Написать регулярное выражение проверки пароля, который должен быть минимум 6 символов, максимум 25, состоять из латинских символов и цифр, может содержать в себе знак подчеркивания
5. Проверить строку на валидность ipv4 адреса

### Document Object Model

1. Вывести задания из раздела “Объекты” в HTML на страницу браузера. Создать формы добавления новых элементов, реализовать возможность удаления и изменения данных.

### TypeScript

1. Все задания которые перечислены выше реализовать на typescript

### Приобретенные скилы

1. Изначальные знания верстки html, css, scss
2. Изначальные знания git (на базовом уровне)
3. Грамотное написание javascript кода (Корректно описание функций, наименования, отступы и стилистика кода)
4. Полное понимание работы javascript совместно с браузером
5. Написание скриптов на Typescript
6. Сборщики проектов: gulp, webpack
7. Знание linux (Основных команд), в том числе работа с пакетным менеджером
8. Работа с npm и его команды

# Проектирование

1. Описание проекта
   1. Парки такси. В парке присутствуют транспортные средства различного класса (Эконом, стандарт, бизнес, универсал, бусик). Транспортное средство может быть собственностью компании, может быть собственностью водителя. У водителя есть рейтинг от 0 до 10. Дополнительная информация о водителях (ФИО, дата рождения, адрес проживания, город и т.д. добавить на свое усмотрение что посчитаете нужным) и аналогично и для автомобилей - это могут быть, номер, марка, модель, год выпуска и другие.
2. При проектировании нужно позаботиться, чтобы следующая функциональность включалась в разработанное АПИ. Добавлять новый автомобиль, добавлять нового водителя, изменять данные по автомобилю и водителю, удалить автомобиль и удалить водителя
3. Подумайте над реализацией безопасности вашего АПИ. Например чтобы доступ мог получить только санкционированный запрос.
4. Продумать все входные данные и выходные каждого АПИ, http коды ответов
5. Разработать для каждого проекта API
6. Написать документацию к каждому проекту
7. Реализовать все API запросы в swagger и postman

### Приобретенные скилы

# Групповые лабораторные работы

## Single page application

1. Сверстать страницу или несколько страниц по предложенному макету. Написать соответствующий javascript код, для реализации работы страниц.
2. Выполнить первое задание на следующем стеке технологий: Typescript, scss, gulp

## Список макетов