#### День 1. #1 Просто запишите

**JS**

Многие из вас и так уже знают что это язык программирования. Один из самых популярных в мире. Вот что говорит нам MDN:

*JavaScript® (часто сокращают до* **JS***) — это лёгкий, интерпретируемый, объектно-ориентированный язык с функциями первого класса, самый известный скриптовый язык для веб-страниц, но также используется во многих не браузерных окружениях. Прототипно-ориентированный, мультипарадигменный язык сценариев, который поддерживает динамический, объектно-ориентированный, императивный и функциональный стили программирования.*

Вы что-то поняли?) Не переживайте, в ближайшее время эти слова станут для вас родными, а пока постарайтесь узнать как можно больше о JavaScript из следующих трех источников (и пусть незнакомые слова вас не пугают):

[learn.javascript.ru](https://learn.javascript.ru/intro#chto-takoe-javascript) [developer.mozilla.org](https://developer.mozilla.org/ru/docs/Web/JavaScript/About_JavaScript) [wikipedia.org](https://ru.wikipedia.org/wiki/JavaScript)

Эти ссылки можно добавить в закладки и обращаться к ним в будущем, если вдруг вы забудете синтаксис или решите освежить в памяти давно пройденную тему

[developer.mozilla.org](https://developer.mozilla.org/ru/docs/Web/JavaScript) [learn.javascript.ru](http://learn.javascript.ru)

**Редактор кода / IDE**

Перед тем как начать писать код, нужно определиться где вы будете его писать. Потратьте достаточно времени чтобы выбрать, установить и изучить вашу будущую среду разработки. С выбором можно и ошибиться - это не страшно, в общем и целом все эти редакторы очень похожи и вам будет просто перейти с одного на другой.

*Термином IDE (Integrated Development Environment, «интегрированная среда разработки») называют мощные редакторы с множеством функций, которые работают в рамках целого проекта. Как видно из названия, это не просто редактор, а нечто большее. IDE загружает проект (который может состоять из множества файлов), позволяет переключаться между файлами, предлагает автодополнение по коду всего проекта (а не только открытого файла), также она интегрирована с системой контроля версий (например, такой как git), средой для тестирования и другими инструментами на уровне всего проекта. Если вы ещё не выбрали себе редактор/IDE, присмотритесь к этим: •* [*Visual Studio Code*](https://code.visualstudio.com/) *(бесплатно). •* [*WebStorm*](http://www.jetbrains.com/webstorm/) *(платно).*

*цитата с* [*learn.javascript.ru*](https://learn.javascript.ru/code-editors#ide)

В чате поделитесь своими мыслями про ваш выбор и расскажите почему решили использовать именно такой Редактор кода / IDE

**Практика**

Сделаем очень простую подготовку к завтрашней задаче - создайте **index.html** и откройте его в браузере. Больше ничего делать не нужно

#### #2 Структура кода

**Пришло время встроить в обычную страницу немного скриптов.**

* У вас уже есть выбранная среда разработки и вы знаете как открыть в браузере обычный **index.html**
* Прочитайте [эту часть справочника](https://learn.javascript.ru/hello-world), используйте тег script и запустите первую программу на JS.
* Она будет очень простой, но уже через пару недель простых шагов вы сможете написать достаточно непростые вещи.

**Структура кода**

После запуска первой программы, самое время перейти к чему-то, что больше одной строчки и познакомиться со структурой кода, а так же, освоить комментарии (но не те что в чате, а те что в коде). Прочитайте [эту статью про структуру кода](https://learn.javascript.ru/structure)

**Экстра бонус для крутышей!**

* В вашем редакторе кода, настройте лайв-сервер и выведите программу из задачи в “живом режиме”.
* Самые прокаченные хакеры могут выделиться еще сильнее - помочь с настройкой сервера всем, у кого это не получится.

#### #3 Терминал и Нода

### Терминал

Мы будем использовать терминал (у кого-то просто командная строка) для установки, настройки и запуска окружения. Нам нужно самое базовое представление и совсем не обязательно заучивать все команды (строго не рекомендуется). На сегодня будет достаточно:

1. Понять что такое Терминал и зачем он нужен (а нам он сейчас на 99% будет нужен только чтобы запускать nodeJS)
2. Научиться переходить из папки в папку (команда ‘cd’)
3. Научить смотреть содержимое папки (команда ‘ls’)
4. Научиться запускать JS файлы с помощью nodeJS (пример: ’node main.js’) - об этом позже

Ссылок не дам, потому что у кого-то bash, у кого-то повершелл, а кто-то вообще на перфокартах. А если серьезно - в вашей повседневной работе никто и никогда не даст вам пошаговых инструкций на все случаи. Да, сейчас вы только учитесь и это может показаться “несправедливо”, но дело в том, что на рабочем месте разработчик вообще постоянно учится, причем, очень часто самостоятельно. Ну, первые года 3 так точно

**Важно:**

1) Удобнее всего запускать терминал прямо в IDE - тогда вам не нужно переключаться между окнами, не нужно прописывать полный путь до файла, так как обычно терминал открывается в нужной вам директории (да, терминал в IDE и терминал вне IDE - это одно и то же)

2) Интерпретатор nodeJS который запускается после установки nodeJS (либо по клику на иконку nodeJS) очень похож на терминал, но это совсем не он, не путайте 2 похожих внешне окна. В интерпретаторе можно вводить выражения на JS, в терминале делать все остальное

### NodeJS

Не переживайте, изучать NodeJS не будем

Из прошлых задач вы уже знаете что браузер является самой популярной средой для исполнения JS, но есть и другие JRE (среды исполнения) и чтоб не бегать за результатами ваших практик в браузер и не выводить результаты в виде алертов, сегодня мы установим NodeJS и будем делать задачи не выходя из IDE Это быстрее и удобнее

В этой задаче кроется 2 других, очень важных для всех кто хочет стать хорошим программистом. Поэтому, я попрошу вас сделать все в точности так, как я описываю это здесь:

* Перейдите на [сайт](https://nodejs.org/ru/) и загрузите LTS версию NodeJS для вашей системы
* После установки, запустите терминал (вы уже знаете как) и попробуйте узнать версию “ноды” командой node --version (может быть команда node -v или node --v) - зависит от вашей системы
* Если вы все сделали правильно, в ответ получите что-то вроде “v14.11.3” (ваша версия может отличаться и это нормально)
* Теперь у вас есть JRE для запуска скриптов без браузера, в терминале перейдите в папку (команда ‘cd’) с заранее созданным main.js (запуская терминал в IDE вы, скорее всего, уже в нужной папке - проверьте наличие вашего файла командой ‘ls’)
* Напишите пару инструкций с выводом любых сообщений console.log(’test’) в вашем main.js
* Запустите ваш main.js командой node main.js (если в терминале вы находитесь в той же папке)

Если вы все сделали правильно, в вашей консоли появится результат работы скрипта main.js

Если у вас что-то не получилось, **повторите** описанное выше, **почитайте ошибки** из вашего терминала и **попробуйте** **решить** эти проблемы самостоятельно, прежде чем обращаться в чат. Эта задача может занять 5 минут или целый день, в зависимости от вашей системы и уровня знаний. Если в течение часа удача вам не улыбнется - смело пишите в чат, вам точно помогут (ведь это будет бонусным заданием для кулхацкеров).

В следующих задачах мы будем использовать и NodeJS и браузер для запуска программ

**Важно:**

1) alert и document не работают в nodeJS - потому что alert относится к браузеру, а document к DOM - ни того, ни другого в nodeJS нет. Подробнее об этом вы узнаете позже, а пока используйте console.log()

2) в 99% случаев понять в чем же проблема можно просто прочитав ошибку в терминале. Если вы не знаете английский - используйте переводчик, со временем большинство слов и ошибок запомнятся и вы будете понимать их на лету)

#### #4 Переменные и типы данных

Начнем с самых базовых концепцией в программировании:

**Переменные**

Любая программа работает с информацией. Переменные нужны чтобы эту информацию где-то хранить. Можно считать что это именованные ящики в которые вы складываете данные и в любой момент можете извлечь их обратно

Переменные являются одним из самых фундаментальных понятий не только в JS, но и в программировании в целом. Эта статья может показаться простой на первый взгляд, при этом, я советую отнестись к ней максимально внимательно и серьезно. Пропустив или не поняв этой информации на 100%, будет очень сложно двигаться дальше

* Прочитайте про [переменные](https://learn.javascript.ru/variables)
* Создайте пару переменных в вашей программе и выведите их в консоль

**Типы данных**

У переменных есть значения, которые в эти самые переменные записаны. Значения переменной в JS всегда относятся к какому-то типу (строка, число, логический тип) Прямо сейчас мы не будем подробно останавливаться на каждом из них, лишь узнаем какими они бывают и поймем саму идею “типизации”

* Прочитайте про [типы](https://learn.javascript.ru/types)
* Создайте переменные с 3-мя разными типами и выведите их в консоль

Ответы на частые вопросы:

* В чем разница между null и undefined? - null - определенное нулевое значение, undefined - неопределенное. Если упростить, null - значение было установлено как пустое. Undefined - значения не установлено.
* Где взять задачи для практики этой теории? Не спешите с практикой, скоро ее станет очень много. Изучив только переменные и типы данных сложно составить осмысленные программы, но без них редко обходятся какие-либо программы вообще.
* Что значит “возвращает”? (пример, оператор возвращает значение) Оператор или функция может возвращать значение. То есть, отдавать вам какие-то данные обратно, отдавать вам результат выполнения в виде какого-то значения.
* Почему IDE ругается на название переменной name? Потому что name является свойством объекта window (но не забивайте себе этим голову прямо сейчас, считайте что это зарезервированное в языке слово)

#### #5 Преобразование типов и базовые операторы

Вы уже знаете про переменные и типы их значений. Теперь стоит понять как эти типы преобразуются в JS Эта тема так же может показаться простой, но в таких вещах ошибаются и программисты с опытом, так как в JS есть свои особенности и понять их интуитивно не получится.

* Прочитайте про [**Преобразование типов данных**](https://learn.javascript.ru/type-conversions)
* Обсудите в чате самые непонятные и нелогичные для вас моменты
* прочитайте про [базовые операторы](https://learn.javascript.ru/operators) (про Побитовые операторы читать строго не рекомендуется)
* напишите программу с разными математическими операторами и разными типами данных. Например сложите строки, строку с числом, число с булевым значением и так далее
* поделитесь вашими результатами в чате (используйте codepen или codesandbox)

*В этой задаче, как и в других на этой неделе, мы постепенно узнаем самые базовые инструкции - это может показаться скучно или слишком просто. Это нужно чтобы уровнять базовые знания и подготовить вас к темпу. Уже на следующей неделе вы напишете простую программу со смыслом и это будет сложнее* 😉

#### #8 Git & GitHub

Сегодня мы продолжим настраивать наши инструменты и в этот раз будет немного интереснее

#### Git

Что это? Распределенная система контроля версий.

А зачем это? Чтобы работать в команде над одними и тем же кодом не мешая друг другу, а еще чтобы записывать все изменения и иметь доступ к предыдущей версии (файла или целого проекта)

А как это работает? Git хранит историю изменения каждого файла, что именно было удалено, что добавлено, кем и когда. Все подписано. Git дает инструменты по организации истории изменений

А еще git - это самый удобный способ поделиться кодом (прислать код на ревью, на проверку). Именно для этого нам и понадобится гит в ближайшее время

#### Установка

Windows: [https://gitforwindows.org](https://gitforwindows.org/)

Mac: brew install git или [скачать пакет](https://git-scm.com/download/mac)

Linux: Вы сами все знаете

#### Регистрация на GitHub

[GitHub](https://github.com) — крупнейший веб-сервис для хостинга IT-проектов, основан на системе контроля версий Git. Зарегистрируйтесь здесь, если вы не сделали этого ранее. Переходите к теории только после успешной установки и регистрации

#### Теория и практика

1. Перейдите в папку с файлом main.js. Можно взять тот, в котором вы делали предыдущие задачи (если у вас его все еще нет - создайте такой файл в любой папке)
2. Чтобы создать git репозиторий, в терминале (находясь в нужной папке) выполните команду git init
3. Результат должен выглядеть так (путь у вас будет свой) Initialized empty Git repository in /Users/username/your\_folder/.git/
4. Добавьте файл main.js в репозиторий. Так мы даем git понять что за изменениями в этом файле надо следить (проводим индексацию изменений) git add main.js
5. Сделайте коммит. Так мы зафиксируем эти изменения в истории.git commit -m "здесь впишите комментарий для вашего коммита"
6. Результат вашего комита будет виден в терминале (он может отличаться от этого примера) [master (root-commit) 911e8c9] мой коммент 1 files changed, 1 insertions(+), 0 deletions(-) create mode 100644 main.js
7. Используйте команду git status, чтобы проверить текущее состояние репозитория (есть ли какие-то изменения и в какой ветке вы находитесь). Если все изменения были успешно добавлены в пункте 5, вы увидите On branch master (или другая ветка) nothing to commit (working directory clean)
8. Добавьте изменения в main.js. Например еще один console.log
9. Снова проверьте статус git status и вы увидите что git знает об этих изменениях и в конце сообщения даже предлагает вам добавить файл или сделать коммит.
10. Сделайте коммит новых изменений (см. пункт 4)
11. Снова проверьте состояние (см. пункт 7) и убедитесь что у вас нет новых изменений

Да, все что вам нужно знать на сегодня это: git init, git add , git commit, git status Если у вас осталось время после выполнения этой практики - изучите эти 4 команды подробнее

### Что делать если что-то пошло не так?

Сначала прочитать сообщение из терминала - в 90% случаев можно понять проблему просто прочитав ошибку (ее можно перевести в любом переводчике). Не все ошибки будут вам понятны, даже на родном языке - это нормально, вы учитесь. Создавайте запросы помощи, задавайте вопросы, но только когда попробуете понять ошибку самостоятельно.

Если вы написали какую-то из команд и далее ваш терминал не реагирует на ваши команды - скорее всего он ждет сообщение для коммита или другой подобный ввод. Самым простым способом будет закрыть терминал и начать сначала. Но можно и отменить действие клавишами (command + C или control + C)

### И самое главное, предупреждая ваши вопросы:

* Какой терминал лучше выбрать и какой круче? Лучшим сейчас будет тот, что уже у вас есть. Не стоит тратить время на выбор оболочек, вы успеете заняться этим пока будете ждать отклики на ваше резюме
* Можно ли писать комменты к коммитам на русском языке? Можно писать коммиты на любом языке, это зависит от проекта
* А чем git отличается от GitHub? git - система контроля версий, GitHub - сервис который на базе этой системы дает вам место, интерфейс и некоторые дополнительные возможности
* А как отменить коммит? Об этом мы поговорим позже. Такая возможность есть

#### #9 Операторы сравнения

Продолжим изучать **базовые концепции в программировании**

Да да, снова простые и не самые интересные вещи, без которых невозможно перейти к серьезным программам и эту тему, так же как и несколько предыдущих, совершенно невозможно игнорировать. Почти все эти операторы знакомы вам еще со школы и, к счастью, в программировании они работают ровно так же, но синтаксис (правила написания) будет немного отличаться. *Каждый понимает что здесь имеется ввиду "a > b”*

Основные сложности возникают при сравнении разных типов данных. На практике хорошим тоном считается сравнивать только одинаковые типы - это проще читается и работает максимально предсказуемо. Для этого разные типы заранее приводят к одному. Но в реальной жизни не все придерживаются хорошего тона и встретить можно что угодно. Вы должны быть к этому готовы

Ваша задача:

* прочитать про [операторы сравнения](https://learn.javascript.ru/comparison)
* попрактиковаться в сравнениях разных типов данных
* проверить друг друга отправив в чат сложносочиненные сравнения (ответы писать не надо, просто решите в уме, а затем проверьте в консоли/терминале)

#### #10 Ветвление и логические операторы

**Условное ветвление**

Это самый первый и основной концепт позволяющий добавить в вашу программу логику. Теперь вы не просто сравниваете и складываете, но еще и решаете по какому пути пойдет ваша программа в зависимости от предопределённых условий. Ветвление добавляет в программы бесконечную гибкость и возможность управлять процессом не зная всех входных данных заранее.

**Если:** Не внимательно читал задачи и пропускал некоторые из них **То:** Так и не стал фронтендером через 7 месяцев

**Если:** Делал всё и точно в срок **То:** Добился поставленной цели

**Задача:**

* прочитать про [**Условное ветвление**](https://learn.javascript.ru/ifelse)
* взять программы из прошлых шагов и добавьте им вариативности, повлиять на результат в зависимости от исходных данных

**Логические операторы**

Логические операторы помогают отличить ложь и истину, взять первое значение или второе.

Эта тема может показаться сложной если до этого вы не писали код.

Эта тема так же является основой основ и требует полного понимания, если вы хотите уверенно двигаться вперед.

Задача:

* прочитать про [Логические операторы](https://learn.javascript.ru/logical-operators)
* использовать их внутри условий

*А вот и ответ на самый популярный вопрос:*

💡

Смысл оператора «вопросительный знак» ? – вернуть то или иное значение, в зависимости от условия. Пожалуйста, используйте его именно для этого. Когда вам нужно выполнить разные ветви кода – используйте if <https://learn.javascript.ru/ifelse>

#### #11 Функции

Шутки в сторону! Мы подошли к функциям и это еще одна фундаментальная часть программирования без которой сложно представить движение вперед. Функции несомненно сложнее всего что мы делали ранее и это второй момент в роадмапе который может вызвать у вас стресс или ступор. Не забывайте про чат, в котором можно задать любой вопрос 😉

**Функции выполняют 2 очень важные задачи:**

* Позволяют использовать один и тот же код в разных местах и с разными параметрами
* Скрывают реализацию. То есть, позволяют использовать какой-то функционал не вдаваясь в подробности о том как он устроен

Вы уже использовали встроенные функции **alert()** и **console.log()** Пришло время научиться писать свои собственные. Задача:

* Внимательно прочитать про [Функции](https://learn.javascript.ru/function-basics)
* Создать простой калькулятор с именем **сalc**, который будет принимать значение a и значение b, а так же идентификатор нужного действия.

**Пример вызова такой функции calc**

сalc(‘add’, 1, 2) - возвращает 3

сalc(‘multi’, 1, 2) - возвращает 2

сalc(’subtract’, 3, 2) - возвращает 1

*Делить пока ничего не надо.*

#### #12 Switch

От большого количества IF может закружится голова, особенно если один из операндов в сравнении всегда повторяется, как здесь

if (a > b) {} if (c > b) {} if (d > b) {}

А еще это может привести к ошибкам и обязательно сделает рефакторинг более трудоемким. В отдельных случаях с оператором Switch можно управлять ветвлением лаконичнее и понятнее чем с IF, но не стоит воспринимать это как полную замену или как лучшую версию. Понимание о том какую конструкцию и в каких случаях использовать придет к вам с опытом

Ваша задача:

* прочитайте про [Switch](https://learn.javascript.ru/switch)
* перепишите ваши ряды if else выражений (в программе calc)

#### #13 Code Review

Настало время сделать своё первое код ревью! И получить немного комментариев в ответ.

Для этого вам нужно:

* принять приглашение в репозиторий Основы (ссылка на него есть в разделе “полезные ссылки” на страницы [Прогресс](http://strada.one/progress) вашего потока)
* клонировать этот репозиторий и перейти на ветку main
* создать ветку вида ваш-ник/practice-1 (artdimitrov/practice-1)
* перейти в эту ветку
* внутри папки клонированного репо создать свою папку (например у меня это ‘artdimitrov’)
* в вашу папку переместить файлы с вашим кодом из прошлой задачи
* сделайте commit изменений и push в remote
* сделать пулл-реквест в ветку main
* через некоторые время во вкладке Pull Requests на GitHub вы обнаружите такие же реквесты от ваших коллег - посмотрите их и оставьте комментарии к их коду (если подобный комментарий уже оставлен - дублировать его не стоит)

#### #15 Циклы

Если **функции** нужны чтобы вызывать один и тот же код в разных местах и с разными параметрами, то **циклы** нужны чтобы повторять одно и то же действие в коде. Циклы, как и функции, очень просто представить в нашей обычной жизни: застегивая пуговицы на рубашке вы повторяете одно и то же для всего набора пуговиц

**Ваша задача:**

* прочитать про [Циклы](https://learn.javascript.ru/while-for) ([другой источник](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/for))
* вывести в консоль числа от 1 до 19 с помощью цикла

*p.s. не забудь выйти из цикла обучения - не более 4 часов в день*

*p.s.s. про Циклы for…of и for…in читать не нужно*

*p.p.s.s. про* [*Прерывание цикла: «break»*](https://learn.javascript.ru/while-for#preryvanie-tsikla-break) *тоже можно не читать*

#### #16 Объекты

В JS без объектов никуда, ведь все здесь на них и построено *(но об этом вы узнаете чуть позже)*

Сегодня мы рассмотрим объекты только как новую структуру данных, не будем сильно вдаваться в остальные детали, пойдем постепенно. Это достаточно сложная тема, на ее понимание требуется куда больше 1 дня и некоторые моменты могут показаться вам не самыми очевидными - не пугайтесь если что-то не понятно прямо сейчас. Мы будем возвращаться к этой теме до самого конца потока и с каждой практической задачей к вам будет приходить больше понимания этой сущности.

Сегодня будет достаточно понять что объект - это тип данных, который используется для хранения сразу нескольких значений

**Ваша задача:**

* внимательно прочитать статью про [Объекты](https://learn.javascript.ru/object)
* добавить 1 объект в ваш калькулятор, с именем operations
* внутри объекта operations перечислить все возможные операции add, multi, sub
* использовать этот объект вместо строк add, multi, sub

Здесь есть небольшая подсказка, но лучше бы вам справиться без нее

const operations = { add: ‘+’, … }

#17 Копирование объекта по ссылке

Вы уже поняли что объекты отличаются от примитивов, теперь стоит понять их самую главную разницу по нахождению в памяти.

но зачем нам сейчас разбираться с памятью?

Согласен, лезть в дебри внутреннего устройства языка нам точно ни к чему, но это одна из тем, без которых ваши программы будут содержать страшные ошибки, исправить которые будет почти невозможно. В прошлой задаче вы имели всего 1 объект и вам не нужно было копировать или изменять его. В реальной работе мы часто копируем объекты, если хотим изменить их значения *(об этом вы узнаете позже)*

Хорошо что на такую сложную тему у вас будет целый день. Прочитайте про [Копирование объектов](https://learn.javascript.ru/object-copy)

#### #18 Контекст и методы объекта

Продолжаем знакомиться с объектами. Сегодня: ключевое слово **this** и понятие "методы объекта" Эта тема не должна показаться сложной если вы хорошо поняли объекты. При этом, это очень важная тема, пропускать которую никак нельзя

[Методы объекта](https://learn.javascript.ru/object-methods)

#### #19 Практика ToDo

Сегодня напишем простой самый простой TODO лист без графического интерфейса.

**Хранилищем будет объект, а имена задач - ключами этого объекта.**

const list = { "create a new practice task": "In Progress", "make a bed": "Done", "write a post": "To Do", }

**Функция changeStatus - будет менять статус задачи**

changeStatus("write a post", "Done")

**Функция addTask - добавляет новую задачу**

addTask('have a walk')

**Функция deleteTask - удаляет задачу**

deleteTask('have a walk')

**Функция showList будет выводить весь список дел в виде**

Todo: "create a new practice task", "make a bed", In Progress: "write a post" Done: -

Создайте список дел, добавьте в него пару задач, поменяйте их статусы несколько раз и выведете результат в консоль

#### #20 Свой проект

Сегодня вы будете заказчиком. Причем, для будущего себя

Вы уже знаете что у вас будут свои проекты к концу марафона. Но не каждый из вас знает что именно это будет. Сегодня отличный шанс сделать свой выбор. Когда выбор будет готов, составьте подробное ТЗ к вашему проекту:

* какую задачу он решает
* как именно будет работать
* опишите весь функционал подробно
* подумайте что стоит сделать в первой версии, а что можно добавить потом

Вы очень удивитесь от двух вещей:

Это будет намного сложнее, чем вам кажется.

Это так сильно изменится спустя 5-6 месяцев

Кидайте в чат то, что у вас получится. Вместе почитаем

#### #22 Числа

Многие из вас уже успели подглядеть эти методы и приемы делая калькулятор. Уметь пользоваться всеми методами необязательно, важно помнить что они существуют и знать что сделать можно, а что нельзя. Из самых часто-используемых можно выделить эти

* [**parseInt и parseFloat**](https://learn.javascript.ru/number#parseint-i-parsefloat)
* [**isNaN**](https://learn.javascript.ru/number#proverka-isfinite-i-isnan)
* [toFixed(n)](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number/toFixed)
* [Math.floor](https://learn.javascript.ru/number#okruglenie)

Прочитайте про [Числа](https://learn.javascript.ru/number)

#### #23 Строки

Вы уже используете строки вместе с другими типами данных, но сегодня вы узнаете их еще лучше и поймете как жонглировать буквами внутри этих строк.

**Ваша задача:**

* Прочитать [эту статью](https://learn.javascript.ru/string)
* Решите все задачи в конце статьи (последнюю можно пропустить)
* Напишите функцию которая преобразует полученную строку в "вертикальный вид" и выводит ее в консоль. Чтобы получилось вот так:

showVerticalMessage('strada') S t r a d a

* Если строка начинается с буквы "s" - нужно вывести эту строку с первой заглавной буквой
* Если строка больше 7 символов - вывести только первые 7

#### #24 Массивы

С типами данных, кажется, разобрались. Самое время перейти к структурам

Вы уже знакомы с циклами, объектами и у вас даже есть опыт использования length и slice() и обращения к элементу по номеру str[0] - кажется что разобраться с массивами вам будет очень и очень просто А вот разобраться в их устройстве и понять как они располагаются в памяти, будет немного сложнее. Но, если вы уже решили задачу со строками, то я уверен что у вас все получится

Ваша задача:

* Прочитать про [массивы](https://learn.javascript.ru/array)

#### #25 Методы массивов

Методы массивов расширяют их возможности и добавляют некоторые элементы функционального программирования.

**Ваша задача:**

Прочитать про [методы](https://learn.javascript.ru/array-methods) массивов.

#### #26 TODO на массивах

Согласитесь, после чтения статьи, вам стало очевидно что для задачи с TODO вам нужен именно массив, а никак не объект? При чем, еще интереснее будет иметь массив объектов, например такой:

const list = [ { name: 'create a post', status: 'In progress', priority: 'low' }, { name: 'test', status: 'Done', priority: 'high' } ]

**Вы знаете что делать :)**

#### #27 Отчет о первом месяце

Сегодня программировать нельзя И даже читать про программирование нельзя. Серьезно. Никакого кода.

Откройте заметки и запишите максимально подробно все что думаете спустя месяц марафона:

* успеваете за основным темпом?
* сами все решаете или вам подсказывают?
* насколько все происходящее здесь совпадает с вашими ожиданиями?
* готовы вот точно так же еще пол года?
* все еще хотите стать программистом?)

А еще, запишите, чему вы научились спустя всего 1 месяц (касается тех, кто пришел без опыта) Просто посмотрите сколько всего поместилось в вас всего за 30 дней. Впереди еще 6 таких же отрезков.

#29 Повторение

Повторите все, что прошли за последний месяц.

Но не прям все все, а те самые темы, на которые ушло больше времени или те, которые не до конца понятны

#### #30 Замыкания

Сегодня начинаются сложные темы, сложнее тех, что были в первый месяц.

Замыкания - это не то, что возникает если учиться по 12 часов в сутки. Это важная концепция в программировании и особенно в JS. Вы уже используете замыкания в своих программах, но теперь пора понять почему это работает и каких ошибок можно избежать в будущем. Эта одна из самых сложных и объемных тем, не пытайтесь понять все и сразу (вряд ли это получится). Мы будем возвращаться к замыканиям раз за разом

Ваша задача: прочитать статью [про замыкания](https://learn.javascript.ru/closure)