



Frontend

# Лекция 1

## Введение

Мартин Комитски



# План на сегодня

- О преподавателях
- О чём курс
- Правила игры
- Основные определения
- Практика
- ДЗ

# Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях



0 нас

# О нас. Преподаватели



Мартин Комитски

[Ответы Mail.ru](#)

Выпускник МГТУ, Технопарк Mail.ru

9 лет опыта, 6.5 года в VK



Дмитрий Зайцев

[Ответы Mail.ru](#)

Выпускник МГТУ

8 лет опыта, 4.5 года в VK

# О нас. Ментор



Тимур

0 курсе



# О курсе. Структура

**Цель курса:** Подготовка к работе в реальной команде на роль frontend разработчика.

Проводился в МФТИ, МИФИ, подобная программа есть в МГТУ.

Курс интерактивный. Будем всячески с вами взаимодействовать, отсиживаться без участия не получится.

- Лекции: 12
  - Всегда есть теория
  - Иногда есть практика
  - Всегда есть ДЗ. Результат работы – Pull Request
- Консультации: 2
  - Можно задать любой интересующий вопрос
  - Можно разобрать ДЗ
  - Можно сдать ДЗ
- Экзамен
  - Разработать небольшое приложение
  - Ревью
  - Вопросы
- Общение в чате – вступайте по ссылке
  - <https://t.me/+UbHWZliOkX4xMWYy>

# О курсе. Модули

- 1 Модуль

- Вводное занятие
- Введение во frontend. Основы JS
- Современные возможности JS API
- Расширенная лекция по CSS
- Консультация 1

- 2 Модуль

- React. Библиотеки, фреймворки, инструменты
- SPA
- Взаимодействие с сервером
- Интерфейсы “ввода”
- Построение сложного интерфейса пользователя
- Авторизация в Web-приложениях
- Консультация 2

- 3 Модуль

- Оптимизация, deploy
- TypeScript
- Экзамен

# 0 курсе. I/O

## Ожидаем, что на входе вы уже:

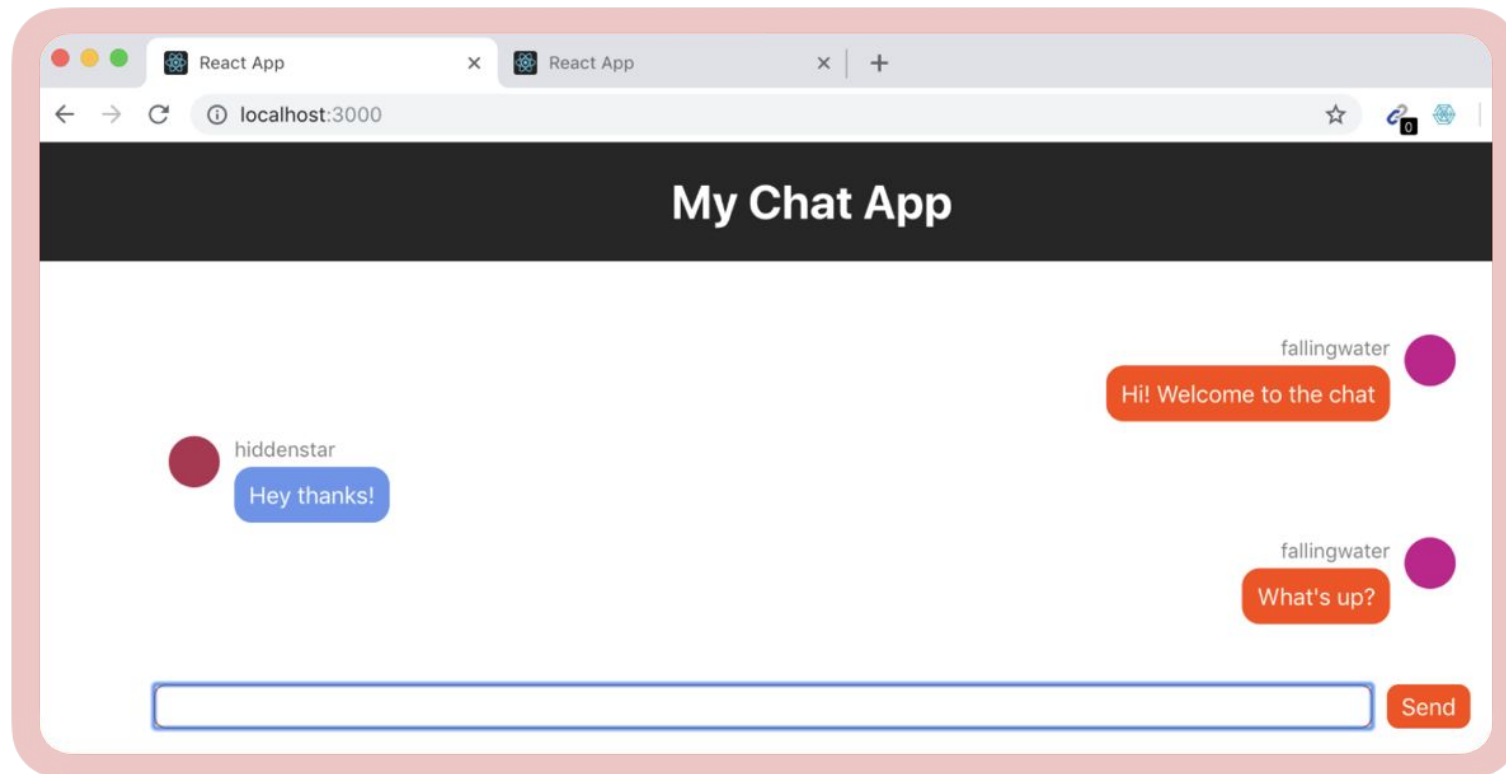
- Умеете использовать git
- Умеете немного программировать
- Имеете свой ПК под управлением:
  - MacOS
  - Ubuntu
  - Windows\*

## Ожидаем, что на выходе вы сможете:

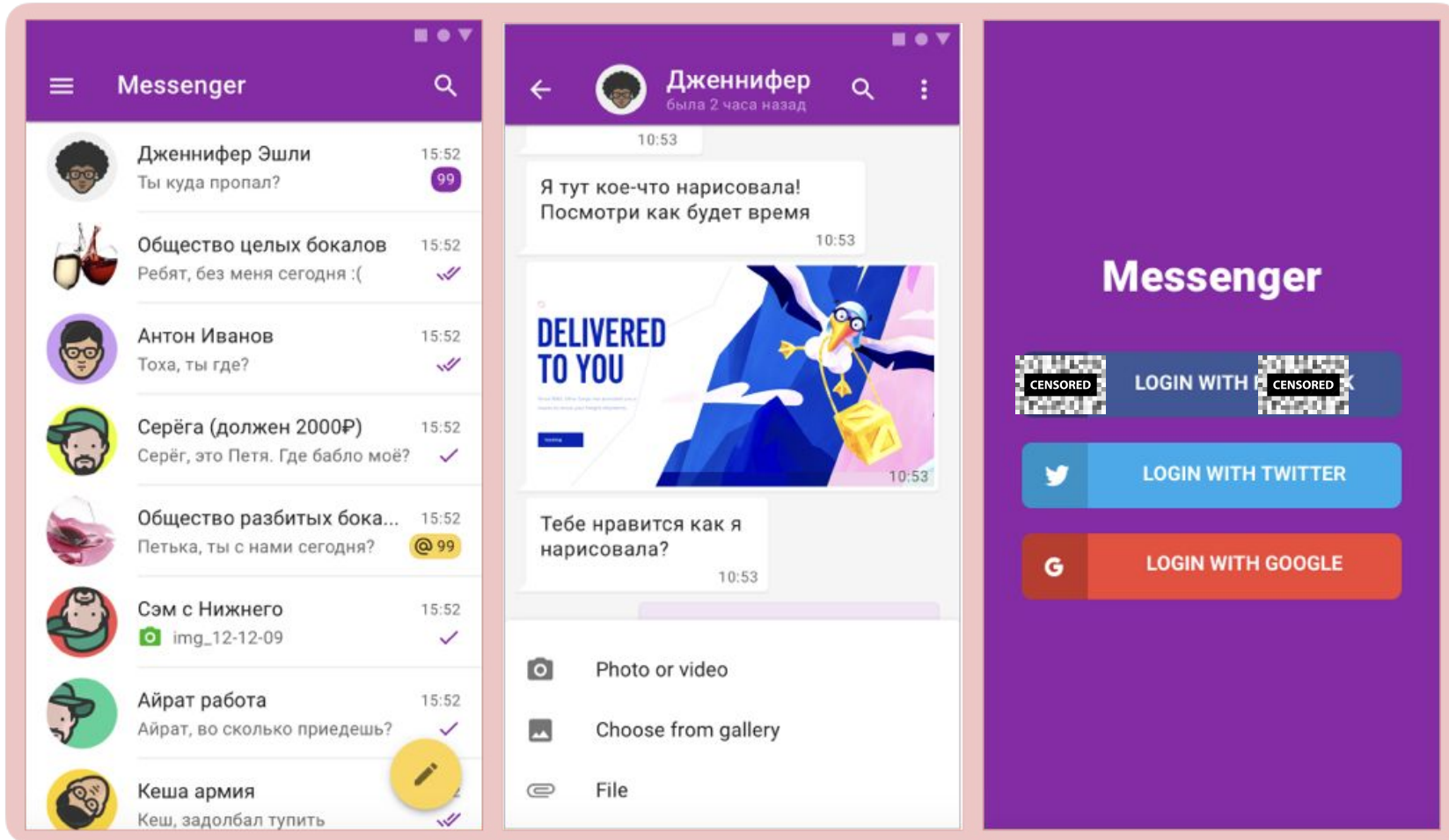
- Разрабатывать сложные web приложения с нуля
  - HTML, JS, CSS
  - React, Redux, TS
  - Jest
  - docker, deploy, CI
  - debug, performance
  - Figma/Zeplin
- Интегрироваться в процесс разработки

# 0 курсе. Проект. Чат

Мы будем разрабатывать приложение “чат”. Вся разработка будет максимально приближена к рабочей атмосфере, будет проходить в github.



# 0 курсе. Проект. Чат



# О курсе. Проект. Чат. Почему?

## Оптимальный набор задач

- Много верстки
- Огромное пространство для идей функций
- Очень много событий, которые надо обработать
- Легко эволюционирует из простого приложения в сложное

# Frontend



# Frontend. Введение

**Frontend** — *публичная часть веб-приложений, с которой непосредственно взаимодействует пользователь.*

Во frontend входят:

- Отображение пользовательского интерфейса
- Функционал, выполняющийся на стороне клиента
- Обработка пользовательских запросов

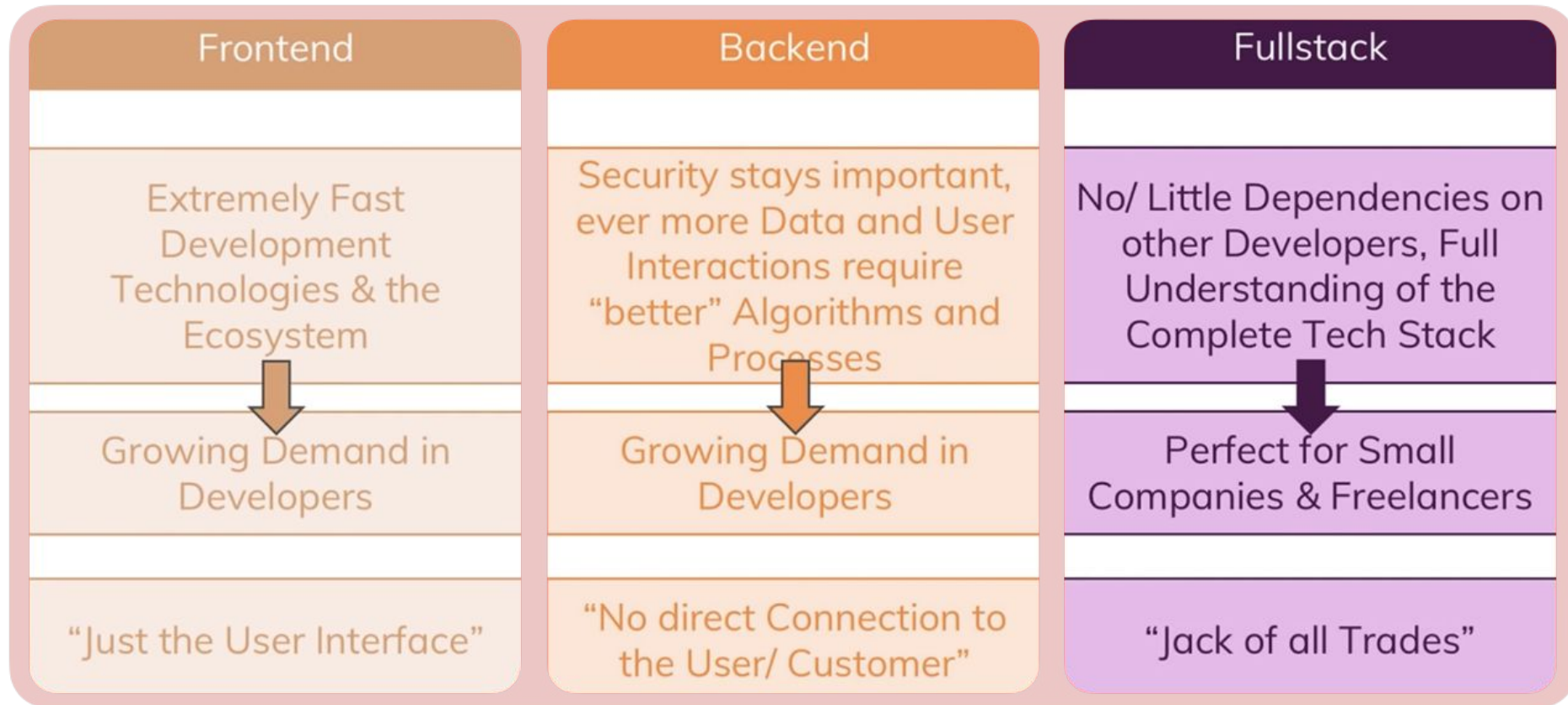
**Frontend разработка** — *работа по созданию публичной части веб-приложения, с которой непосредственно контактирует пользователь и функционала, который выполняется на стороне клиента.*



# Frontend. Введение. Почему?

- Результат работы виден сразу
- Очень быстрое развитие экосистемы
- Топ-3 язык программирования
- Большое сообщество
- Низкий порог вхождения (иногда встречается плохой код 🤔)
- Высокий спрос специалистов на рынке труда => Высокий уровень ЗП

# Frontend. Введение. Роли в web-разработке



# Frontend. Введение

## Технологии

- HTML
- CSS
- JavaScript
- CSS пре-, пост-процессоры (Sass, Stylus, PostCSS)
- JavaScript библиотеки (lodash, underscore, ramda)
- JavaScript ui-библиотеки/фреймворки (react, vue, angular, svelte)
- Менеджеры пакетов, сборщики, прочие инструменты (npm, webpack, jest)

# Frontend. Введение

## С чем можно работать

- UI
- Написание JS логики и CSS стилей для UI компонентов
- Анимации, переходы
- Формы и валидация пользовательского ввода
- Обработка пользовательских событий
- Алгоритмы
- Общение с backend
- Использование продвинутых API
- Реализация продвинутых UX стратегий (PWA, Web-RTC)
- Использование графических движков (Unity, Pixi.js и др.)

# Frontend. Введение

**Node.js** — среда для выполнения JS кода. Нужна для работы с инструментами, фреймворками/библиотеками (*webpack, express, koa, react, vue*).

**npm, yarn** — пакетные менеджеры для *nodejs* модулей.

# Frontend. Введение

**Линтеры** — набор программ и инструментов для проверки кода на соответствие определенным правилам.

Основные: eslint, stylelint

# Frontend. Введение

**CI/CD** — *continuous integration/continuous delivery*

Набор техник для реализации непрерывного цикла разработки и слияния нового кода в релизную версию продукта.

# Frontend. Введение

**Docker** — инструмент для запуска приложений в изолированном окружении (в контейнере).



# Frontend. Введение

**Travis** — популярный сервис для сборки проектов.

Легкая интеграция с сервисом github.

Аналоги: jenkins, gitlab-ci, circleci

# Frontend. Введение

**Генератор проектов** — *инструмент для быстрой генерации проекта.*

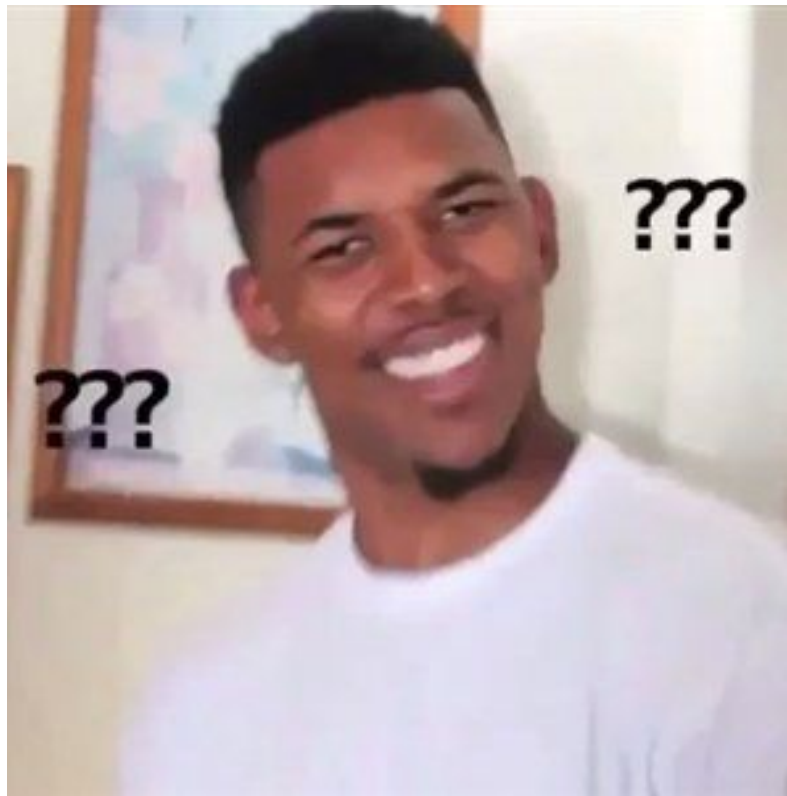
Примеры генераторов:

**create-react-app, yeoman, slush**

Другие названия: "скаффолдер", "бутстраппер", "шаблон проекта".

# Frontend. Введение?

Вопросы?





Перерыв! (10 минут)

Препо (с)

# Практика и ДЗ

# Что делаем

- Запускаем ваш любимый терминал
- Устанавливаем среду исполнения `nodejs` и пакетный менеджер `npm` (можно поставить пакет `nvm` для быстрой смены версий обоих инструментов)

1. `npm -v`
2. `node -v`

# Что делаем

1. `npm install -g yo generator-track-mail`
2. `mkdir my-project && cd $_` # Позже вы создадите репозиторий, просто перенесите всё содержимое папки в него
3. `yo track-mail:functions`

# Что делаем

```
1.  npm init
2.  npm install --save-dev jest
3.  npm install --save-dev babel-jest @babel/core @babel/preset-env
4.
5.  # Добавить в package.json следующий код
6.    "scripts": {
7.      "test": "jest"
8.    },
9.    "jest": {
10.     "transform": {
11.       "^.+\\.jsx?$": "babel-jest"
12.     }
13.   },
14.   "babel": {
15.     "presets": ["@babel/preset-env"]
16.   },
17.
18.  npm test
```



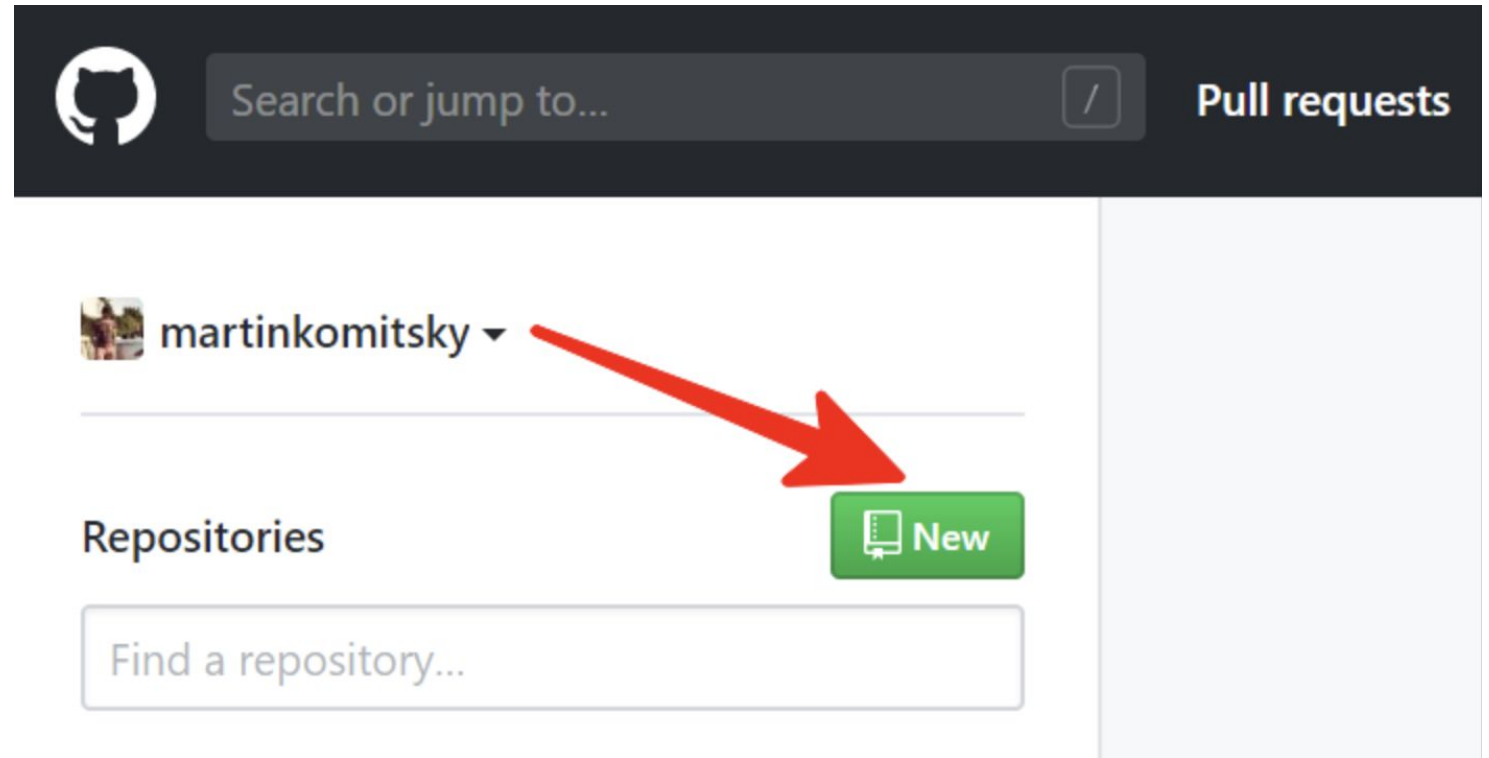
ДЗ

## ДЗ. Как сдавать

- Выполнить локально задачи
- Закоммитить изменения в ветку ``dz#``, где “#” – номер домашнего задания
- Запустить ветку в свой github
- Поставить ПР с ``dz#`` на ``master``
- Запросить ревью ПР у текущего преподавателя

## ДЗ. Инструкция. 1. Новый репозиторий

- Заходим на GitHub
- Создаем новый репозиторий
- [Полная инструкция](#)



## Д3. Инструкция. 2. Правильное имя

Присваиваем имя репозиторию


**Шаблон имени:**

- YYYY-HALF\_YEAR-VK-EDU-FS-Frontend-N-LAST\_NAME
- YYYY – год
- HALF\_YEAR – половина года. 1, если сейчас месяц янв-июн, 2, если июл-дек
- N – первая буква имени
- LAST\_NAME – фамилия

**Пример:** 2022-2-VK-EDU-FS-Frontend-M-Komitsky

## ДЗ. Инструкция. 2. Правильное имя


**Owner \*** **Repository name \***


 martinkomitsky / 2022-2-VK-EDU-FS-Frontend-l ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-happiness?](#)

**Description (optional)**

Учебный проект

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.


☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)


**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

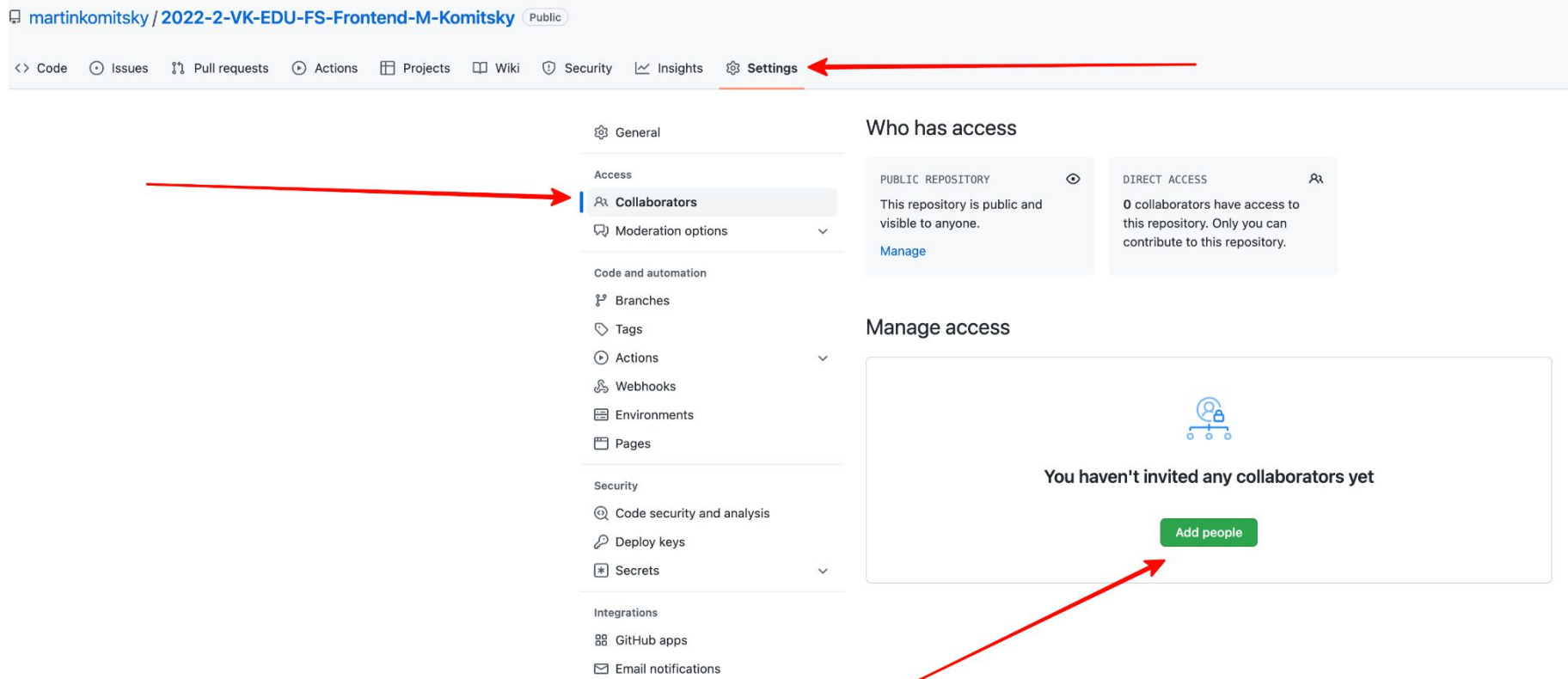
 You are creating a public repository in your personal account.

**Create repository**

# ДЗ. Инструкция. 3. Коллабораторы

## Добавить преподавателей

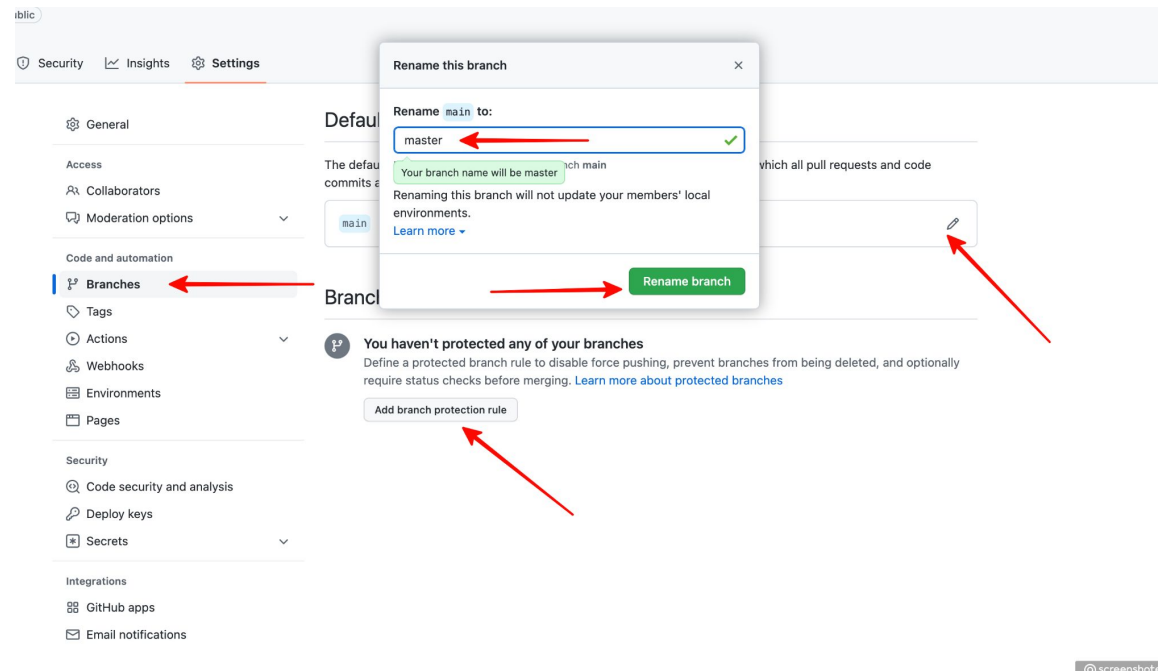
- Заходим в настройки репозитория (/settings/collaboration)
- Добавляем martinkomitsky и haseprogram в collaborators



# ДЗ. Инструкция. 4. Создание правила

## Создание правила для веток

- Заходим в настройки веток (branches)
- Создаем новое правило
- Защищаем ветку master от пуша
- Требуем обязательный апрув от одного ревьюера для мержа ПР



# ДЗ. Инструкция. 5. Настройка правила

The screenshot shows the GitHub 'Branches' settings page for configuring a new branch protection rule. The left sidebar contains navigation links: Branches, Tags, Actions, Webhooks, Environments, Pages, Security, Code security and analysis, Deploy keys, Secrets, Integrations, GitHub apps, and Email notifications. The main content area is titled 'Branch name pattern' and 'Protect matching branches'. The 'Branch name pattern' field contains 'master'. Under 'Protect matching branches', several options are checked: 'Require a pull request before merging', 'Require approvals' (with 'Required number of approvals before merging' set to 1), and 'Dismiss stale pull request approvals when new commits are pushed'. Other unchecked options include 'Require review from Code Owners', 'Require status checks to pass before merging', 'Require conversation resolution before merging', 'Require signed commits', 'Require linear history', 'Require deployments to succeed before merging', and 'Do not allow bypassing the above settings'. At the bottom, under 'Rules applied to everyone including administrators', 'Allow force pushes' and 'Allow deletions' are unchecked. A green 'Create' button is at the bottom left, and a '@screenshoter' watermark is at the bottom right. Red arrows highlight the 'master' pattern, the 'Require a pull request before merging' checkbox, the 'Require approvals' checkbox, the 'Required number of approvals before merging' dropdown, the 'Dismiss stale pull request approvals...' checkbox, and the 'Create' button.

Branches

- Tags
- Actions
- Webhooks
- Environments
- Pages

Security

- Code security and analysis
- Deploy keys
- Secrets

Integrations

- GitHub apps
- Email notifications

Branch name pattern \*

master

Protect matching branches

- ☒ **Require a pull request before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.
- ☒ **Require approvals**  
When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.  
Required number of approvals before merging: 1
- ☒ **Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.
- ☐ **Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.
- ☐ **Require status checks to pass before merging**  
Choose which **status checks** must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.
- ☐ **Require conversation resolution before merging**  
When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more.](#)
- ☐ **Require signed commits**  
Commits pushed to matching branches must have verified signatures.
- ☐ **Require linear history**  
Prevent merge commits from being pushed to matching branches.
- ☐ **Require deployments to succeed before merging**  
Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.
- ☐ **Do not allow bypassing the above settings**  
The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

Rules applied to everyone including administrators

- ☐ **Allow force pushes**  
Permit force pushes for all users with push access.
- ☐ **Allow deletions**  
Allow users with push access to delete matching branches.

Create

@screenshoter



## ДЗ. Инструкция. 6. Шаблон для ПР

### Создаем шаблон для пулл реквестов

- Создаем файл с именем pull\_request\_template.md в корне проекта
- Содержимое должно быть следующим:
- # Домашнее задание №
- 
- Прошу @martinkomitsky или @haseprogram проверить его.
- 
- Что было сделано:
- \*
- \*
- \*

# ДЗ. Инструкция. 7. Сдача ДЗ

## Сдаем ДЗ на проверку

- После выполнения домашнего задания, создаем пр в ветку master
- Добавляем martinkomitsky и haseprogram в поле reviewers
- Добавляем того, кто выдал домашнее задание (лектора конкретной лекции) в поле assignee
- В теме обязательно пишем номер ДЗ, в описании опционально пишем то, что сделано

# ДЗ. Инструкция. 7. Сдача ДЗ

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master ← compare: devel ✓ Able to merge. These branches can be automatically merged.

ДЗ 1

Write Preview

H B I

# Домашнее задание №1  
Прошу @martinkomitsky или @haseprogram проверить его.  
  
Что было сделано:  
\* Добавлена верстка профиля  
\* Исправлены баги  
\* Улучшена производительность

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviewers

HaseProgram

martinkomitsky

At least 1 approving review is required to merge this pull request.

Assignees

martinkomitsky

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Use [Closing keywords](#) in the description to automatically close issues.

## Д3. Инструкция. 8. Проверка и правки

### **Ожидаем проверки и вносим правки**

- Жмем на большую зеленую кнопку и ждем комментариев
- Если все выполнено корректно - выполняется мердж
- Если есть недочеты - будут оставлены замечания, которые надо исправить и запустить в текущий PR
- Мы видим правки, повторяем процесс ревью
- Пересоздавать PR не нужно

## ДЗ. Инструкция. 9. Правила сдачи ДЗ

- Для всех ДЗ один репозиторий
- В одном PR сдается только одно ДЗ
- Каждое ДЗ делается в отдельной ветке, чтобы избежать возможных конфликтов при мерже
- Каждая ветка, из которой делается PR, должна быть синхронизирована с master

## ДЗ. Критерий оценки

- Для того, чтобы успешно сдать домашнее задание без применения штрафов (снятие 30% баллов) - нужно вовремя заблаговременно создать PR
- Срок сдачи каждого задания - 2 недели с момента его выдачи на лекции, но не позднее 23:59:59 дня перед лекцией
- Если вы уложились с решением ДЗ в 2 недели, то получаете к максимальным 10 баллам еще дополнительных 2 за "скорость"
- Если вы не уложились в 2 недели, но сдали ДЗ на семинаре, то вы получаете максимально 10 баллов
- Если вы не сдали ДЗ на семинаре текущего модуля, то применяется штраф 30% от максимальных 10 баллов, что делает максимальную оценку после штрафа равной 7
- Максимальной оценкой оценивается решение, которое было выполнено правильно с первого раза
- Если были допущены сильные недочеты, то баллы снижаются в зависимости от тяжести положения

# Просьба

- Заполнить профиль реальными именами и фото на портале/в telegram
- Следить за блогом на портале
- Задавать вопросы в чат
- Зазубрить инструкцию по сдаче ДЗ
- Понимать, что жесткие требования обусловлены желанием максимально быстро вогнать в атмосферу реальной разработки, которая будет на стажировке

# Полезные ссылки

- Читать:
  - [YDKJS: Getting Started](#)
  - <https://ohmyz.sh/>
- Смотреть:
  - [The Weird History Of JavaScript](#)
  - [JavaScript: How It's Made](#)
  - [How to pronounce programming languages](#)



# Домашнее задание №1

1. В своем проекте выполнить `yo track-mail:functions`
2. Выполнить задачи в файлах:
3. `convertBytesToHuman.js`, `correctSentence.js`, `nonUniqueElements.js`
4. Написать тесты в соответствующих файлах (`*.test.js`)
5. Освоиться с терминалом, `shell`, настроить тему и инструменты `dotfiles`

Расширенное описание задания, подсказки, а также презентации с лекций всегда есть в репозитории.

Срок сдачи

*7 октября*

# ДЗ?

Вопросы?



# Спасибо за внимание!



# Пока!

Присоединяйтесь к сообществам про образование в VK

- [VK Джуниор](#)
- [VK Образование](#)

