Лекция 1.
Не введение

ЦРИТО
Центр Развития
ИТ-Образования

МФТИ

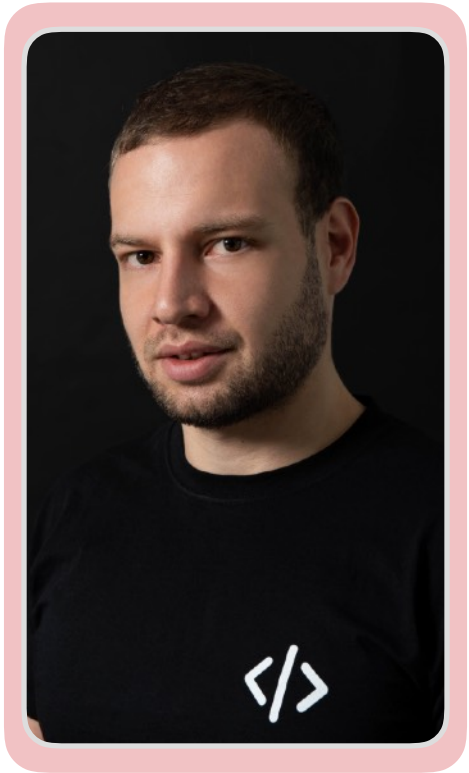
@ mail.ru
group

Важно

1. Отметиться на портале
2. Заполнить анкету
3. Оставить обратную связь

Класс!

Это я



Алексей
Опалев

@a.opalev
Медиапроекты@Mail.RU

Глоссарий

Фронтенд

Публичная часть веб-приложений, с которой непосредственно взаимодействует пользователь.

Во фронтенд входят: отображение пользовательского интерфейса, функционал, выполняющийся на стороне клиента, и обработка пользовательских запросов.

Глоссарий

Фронтенд разработка

Работа по созданию публичной части веб-приложения, с которой непосредственно контактирует пользователь и функционала, который выполняется на стороне клиента.

Глоссарий

nodejs

Среда для выполнения JS кода.
Нужен для работы с фреймворками/библиотеками (react, vue, express).

Глоссарий

npm

Пакетный менеджер для nodejs модулей.

Глоссарий

Линтеры

Набор программ и инструментов для проверки кода на соответствие определенным правилам.

Основные: eslint, stylelint

Глоссарий

CI/CD

Continious integration/continious delivery

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

Глоссарий

Docker

Инструмент для запуска приложений в изолированном окружении (в контейнере).

Глоссарий

Travis

Популярный сервис для сборки проектов.
Легкая интеграция с сервисом github.

Аналоги: jenkins, gitlab-ci, circleci

Глоссарий

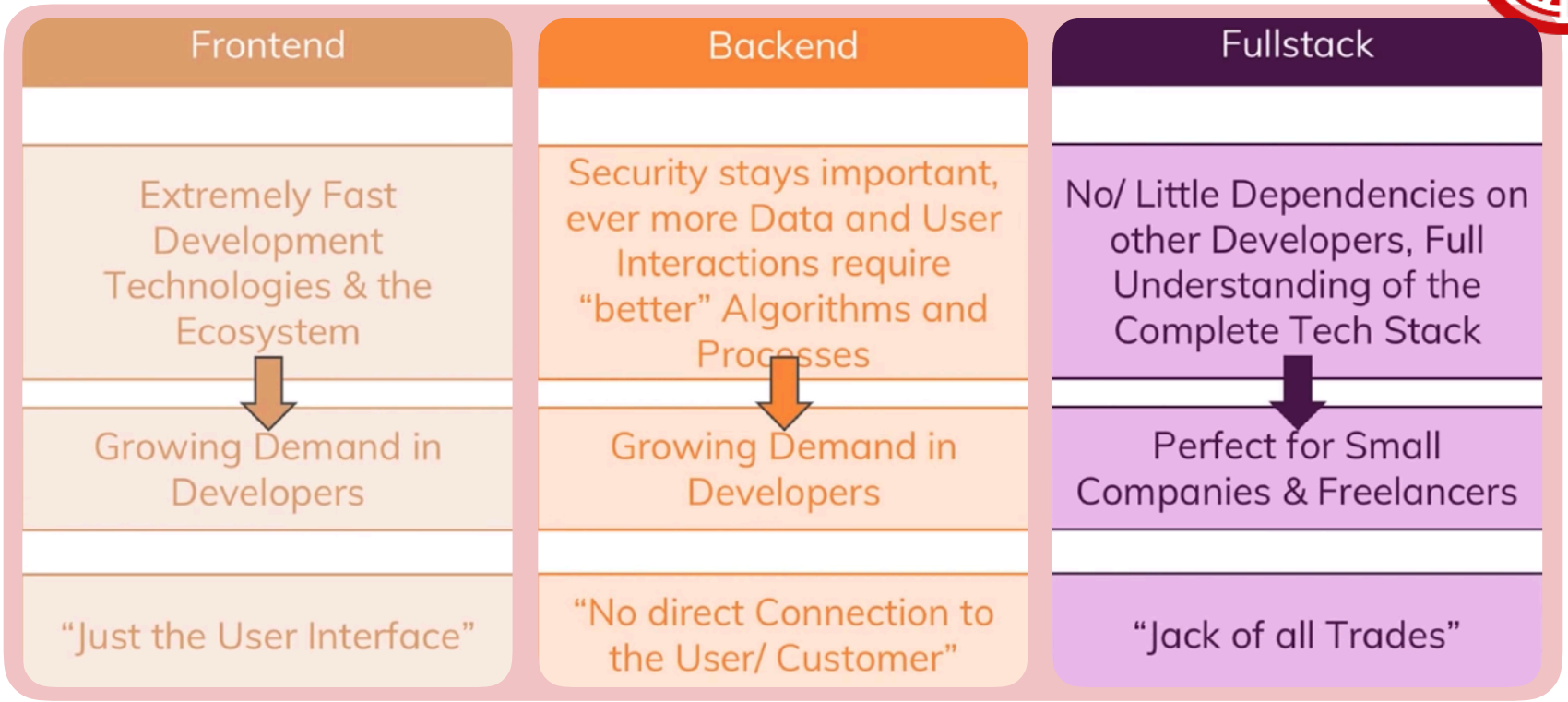
Генератор проектов

Программа/инструмент для генерации проекта.

Примеры генераторов:
create-react-app, yeoman, slush

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

Роли в веб разработке



Почему фронтенд

- Сразу видно результат работы
- Очень быстрое развитие экосистемы
- Большое сообщество
- Низкий порог вхождения [много плохого кода :-( ]

Технологии

1. HTML
2. CSS
3. JavaScript
4. CSS пре, пост-процессоры (Sass, Stylus, PostCSS)
5. JavaScript библиотеки (lodash, underscore, rambda)
6. JavaScript frameworks (react, angular, vue, svelte)
7. Сборщики, менеджеры пакетов (npm, webpack)

С чем работать

1. UI
2. Написание JS логики и CSS стилей для UI компонентов
3. Формы и валидация пользовательского ввода
4. Общение с бэкендом
5. Реализация продвинутых UX стратегий (PWA, RTU)

Цель курса

Подготовка к работе в реальной команде на роль фронтенд разработчика.

- **10 занятий, 2 семинара**
- **ДЗ после каждого занятия**
- **Результат работы - pull request**
- **Экзамен [защита своего проекта, решений, кода]**

Блоки лекций

- HTML - CSS - JavaScript 
- React - Single Page Application - Forms 
- Server - Complex Interfaces - Authorization 

Вопросы на экзамене [примеры]

Что такое

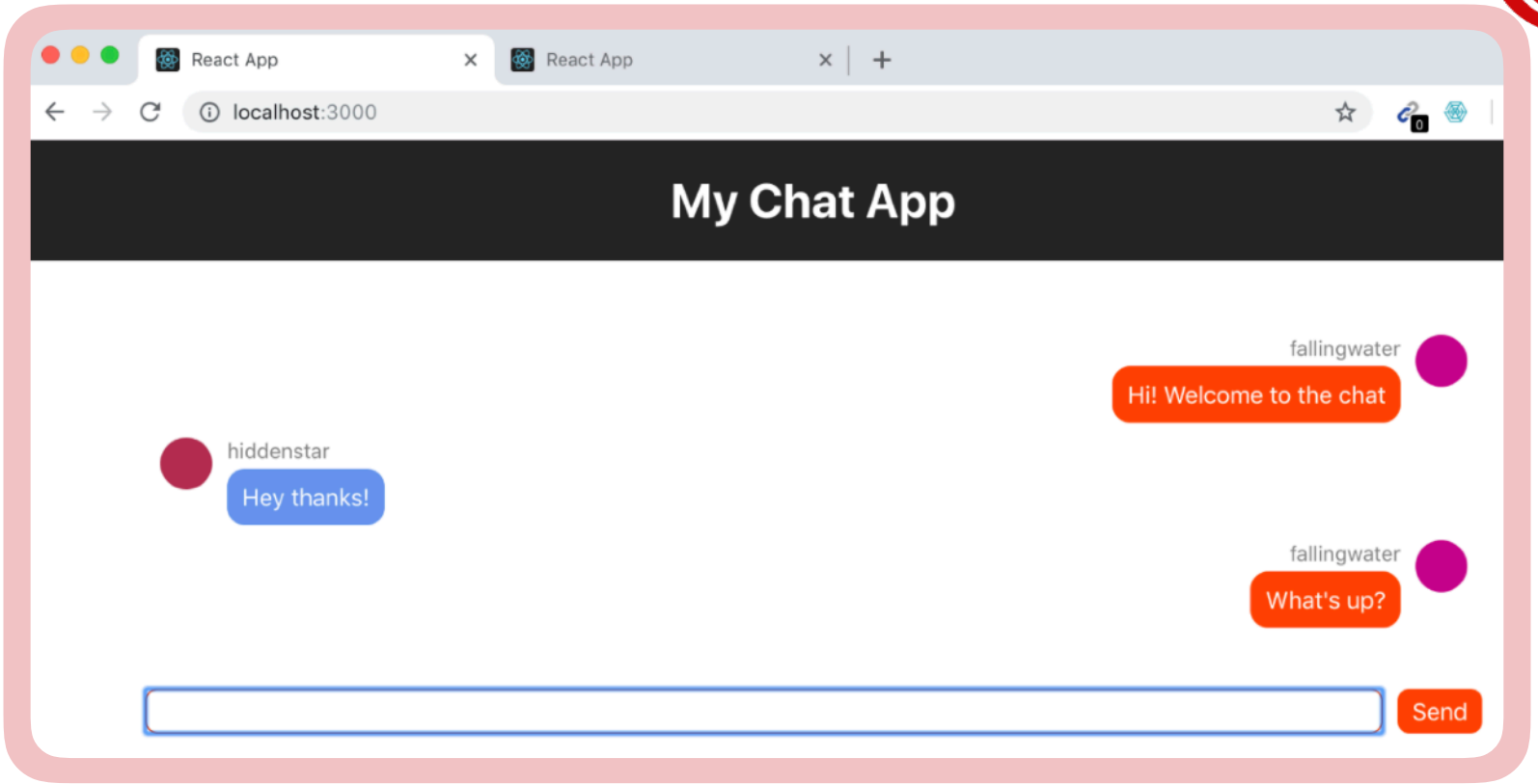
1. devtools
2. javascript
3. async
4. transport
5. I/O
6. React/Angular/Vue
7. jQuery

Вопросы на экзамене [примеры]

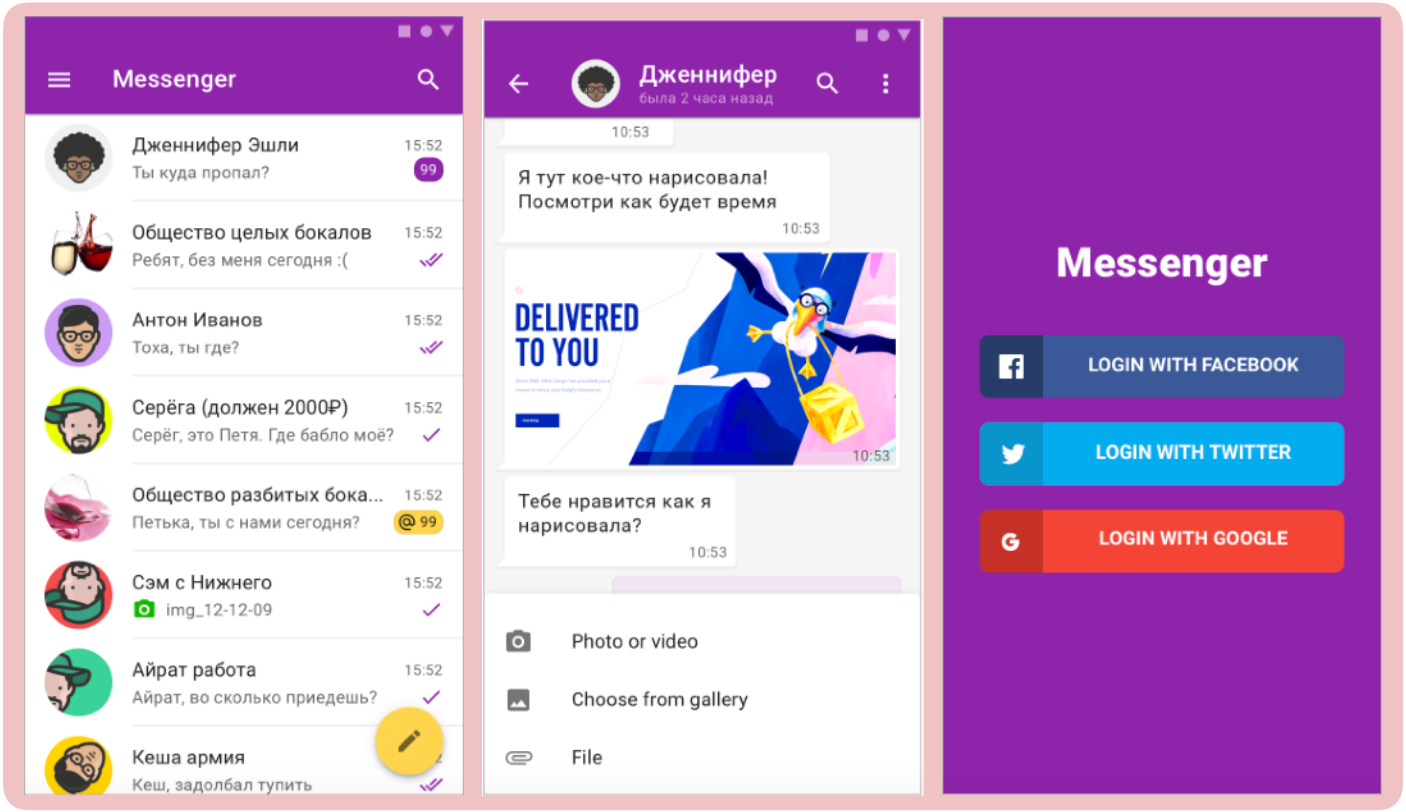
Как происходит

1. Загрузка страницы
2. Отладка приложения
3. Деплой приложения
4. Коммуникация между клиентом и сервером

Приложение – Чат



Приложение – Чат



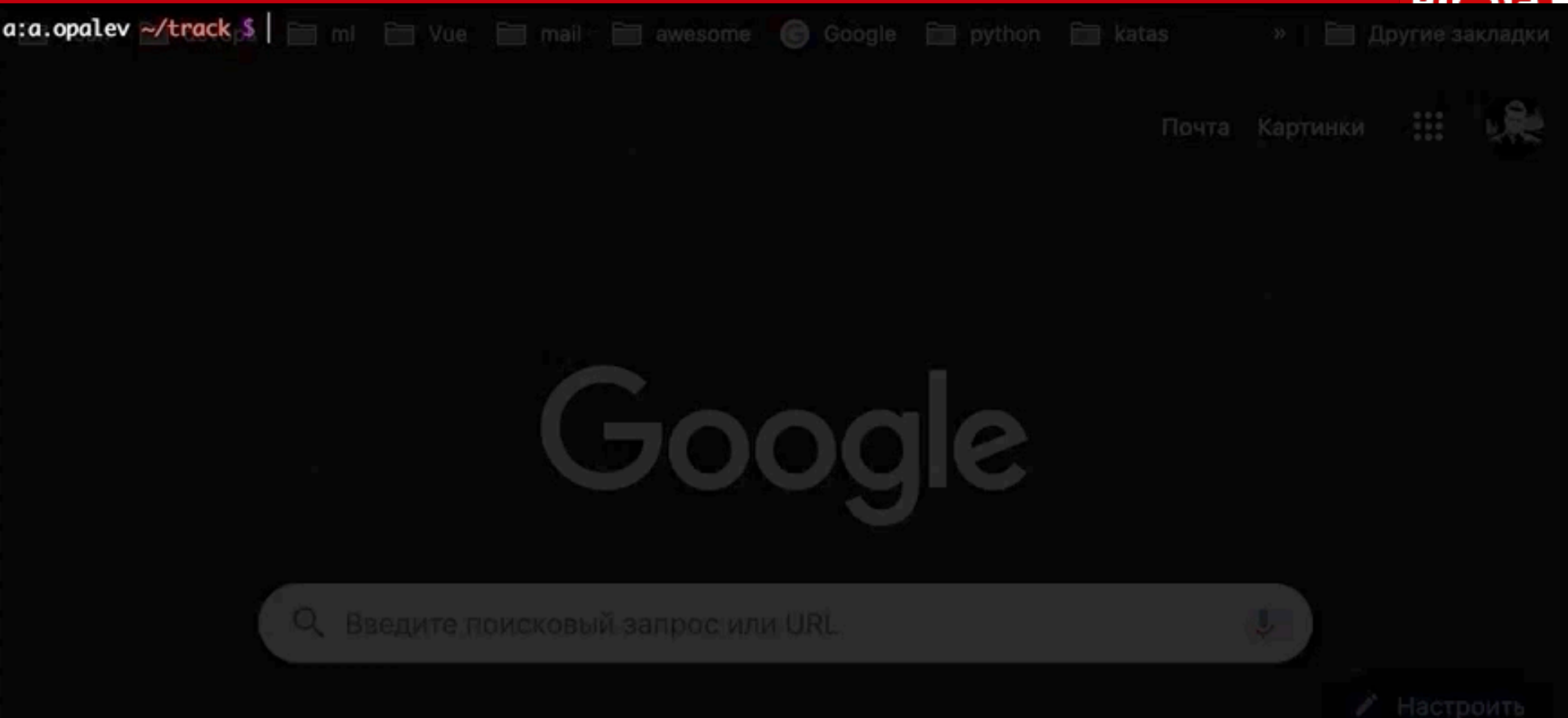
Оптимальный набор задач

- Много верстки
- Очень много событий, которые надо обработать
- Легко эволюционирует из простого приложения в сложное

Создаем проект

```
1. npm install -g yo
2. npm install -g generator-track-mail
3. mkdir project && cd $_
4. yo track-mail
```

Что происходит [mkdir]



Что происходит [docker-compose]

```
Compiled successfully!

You can now view track-mail-2019-a-opalev in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.1.67:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.

^C
a:a.opalev ~/track/proj $
```

Что происходит [open]

```
Compiled successfully!

You can now view track-mail-2019-a-opalev in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.1.67:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.

^C
a:a.opalev ~/track/proj $ docker-compose up -d
Creating network "proj_webnet" with the default driver
Creating proj_frontend_1 ... done
a:a.opalev ~/track/proj $
```

От сложного

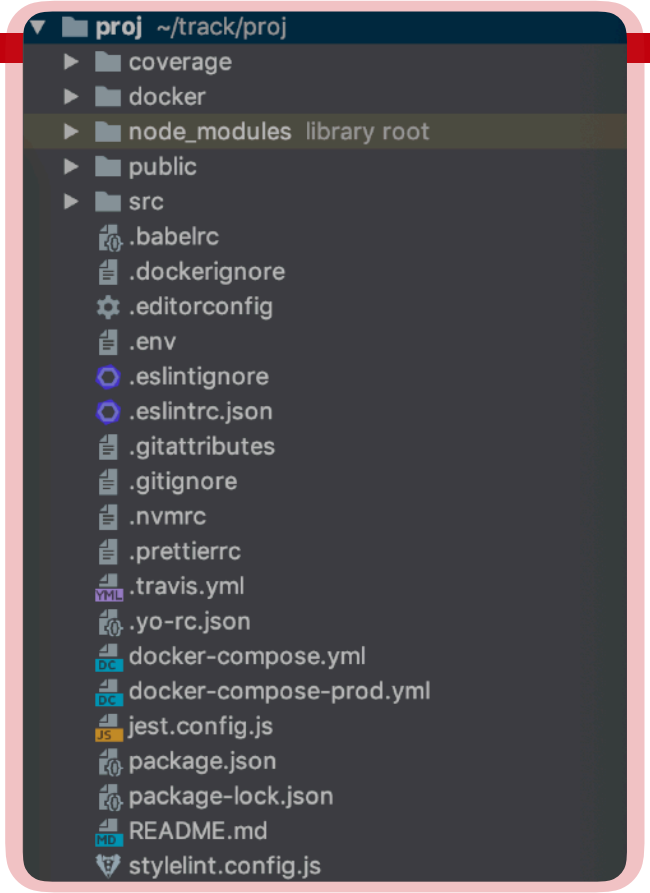
К сложному. Но чуть-чуть проще.

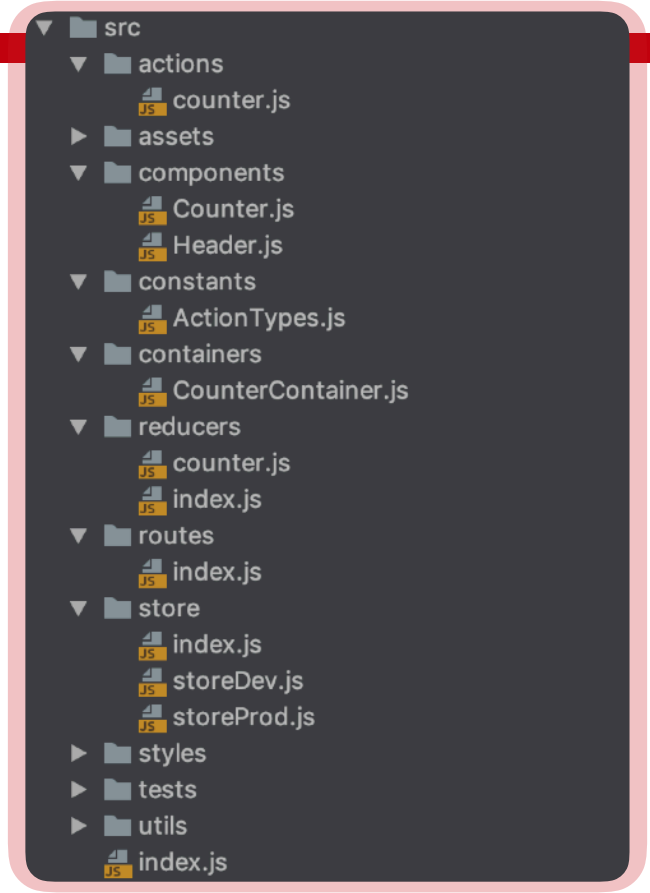
Что содержится в проекте

Веб приложение

1. готовое приложение [react, redux, routes]
2. линтеры [eslinter, stylelint, prettier]
3. docker
4. travis
5. git hooks
6. тесты [jest]

Структура проекта





Со своим веб приложением

Залить в свой гитхаб. [Потребуется для выполнения дз]

Попытаться сломать приложение.

Попытаться починить приложение.

Разобраться в файлах конфигураций (АКА конфиги).

Популярная #3

Файлоориентированная.

```
api/
  APIUtils.js
  APIUtils.test.js
  ProfileAPI.js
  UserAPI.js
components/
  Avatar.js
  Avatar.css
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  Profile.js
  ProfileHeader.js
  ProfileHeader.css
```

Структуры проектов

Популярная #2

Фичеориентированная.

```
common/
  Avatar.js
  Avatar.css
  APIUtils.js
  APIUtils.test.js
feed/
  index.js
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  FeedAPI.js
profile/
  index.js
  Profile.js
  ProfileHeader.js
```

Структуры проектов

Популярная #1

Атомный дизайн.
http://bradfrost.com/blog/post/atomic-web-design

```
├── src/
│   ├── _settings/
│   │   └── variables.css
│   ├── atoms
│   │   ├── button/
│   │   │   ├── index.js
│   │   │   ├── stories.js
│   │   │   └── style.css
│   │   └── *
│   ├── molecules/
│   │   └── *
│   ├── organisms/
│   │   └── *
│   └── templates/
│       └── default.css
└── pages/
    ├── home/
    └── section/
```

Правильная

move files around until they feel right © Dan Abramov

Функции и тесты

В своем проекте выполнить `yo track-mail:functions`
Выполнить задачи в файлах:
convertBytesToHuman.js correctSentence.js nonUniqueElements.js
Написать тесты в соответствующих файлах [.test.js]
Запустить команду docker-compose exec frontend npm run test

Домашнее задание

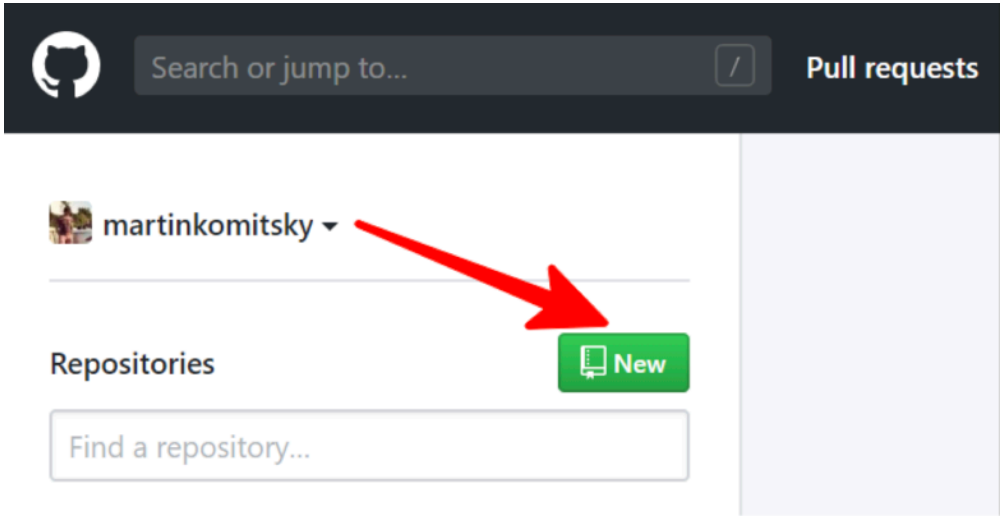
Как сдавать дз

1. Выполнить локально задачи
2. Закоммитить изменения в ветке `devel`
3. Запушить ветку в свой гитхаб
4. Поставить ПР с `devel` на `master`
5. Запросить ревью ПР у меня [https://github.com/chexex]

ДЗ. Инструкция. 1. Новый репозиторий

Создаем репозиторий

1. Заходим на GitHub
2. Создаем новый репозиторий



ДЗ. Инструкция. 2. Правильное имя.

Присваиваем имя репозиторию

Шаблон имени:

`YYYY-HALF_YEAR-TRACK-Frontend-N-LAST_NAME`

`YYYY` - год

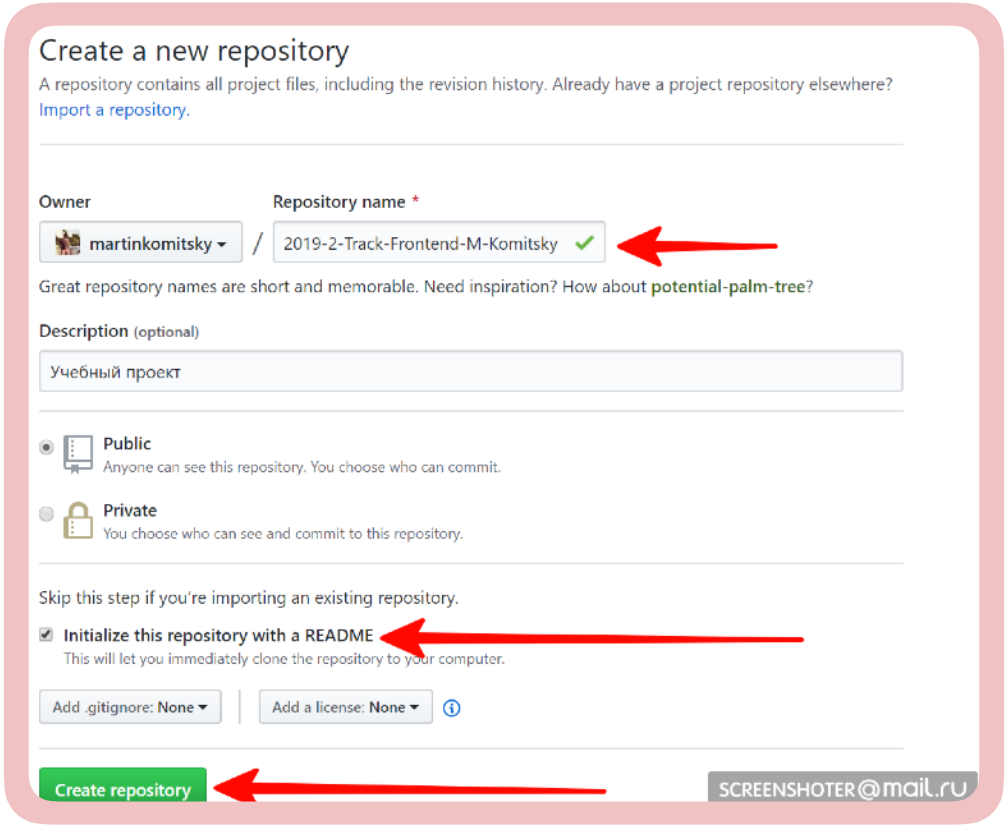
`HALF_YEAR` - половина года. `1`, если сейчас `янв-июн`, `2`, если `июл-дек`

`N` - первая буква имени

`LAST_NAME` - фамилия

Пример: **2019-2-Track-Frontend-A-Opalev**

ДЗ. Инструкция. 2. Правильное имя.

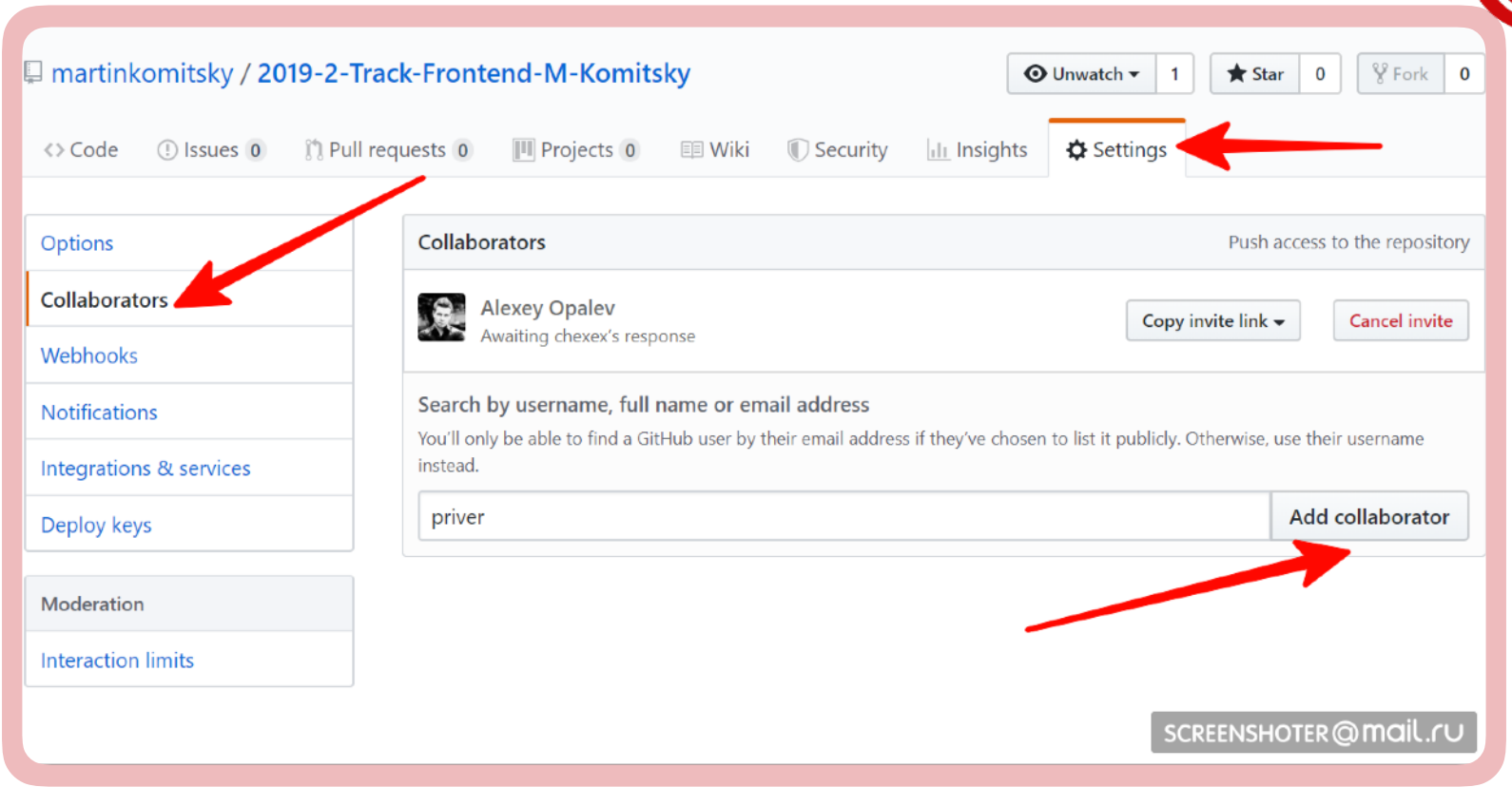


ДЗ. Инструкция. 3. Коллабораторы

Добавить преподавателей

1. Заходим в настройки репозитория (/settings/collaboration)
2. Добавляем `martinkomitsky`, `chexex` и `priver` в collaborators

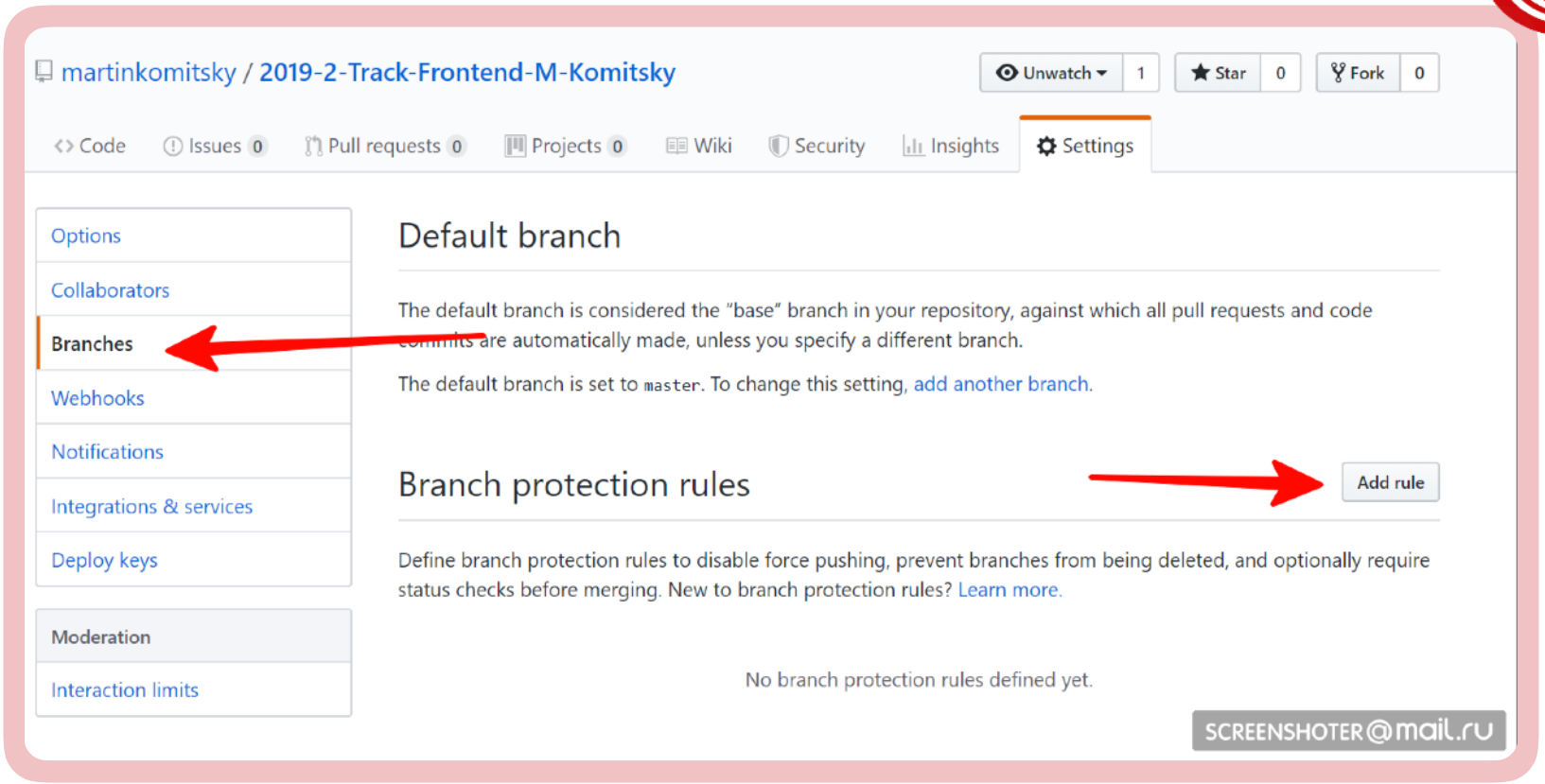
ДЗ. Инструкция. 3. Коллабораторы



Создание правила для веток

1. Заходим в настройки веток (branches)
2. Создаем новое правило

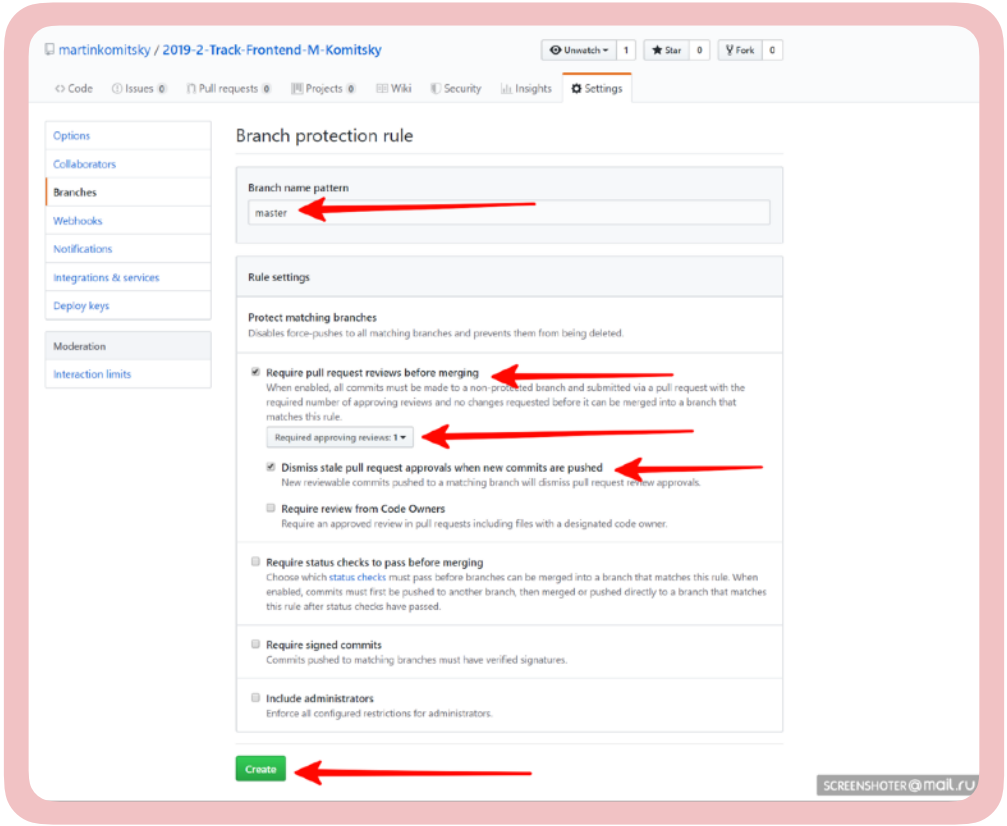
ДЗ. Инструкция. 4. Создание правила



Настройка правила для веток

1. Защищаем ветку `master` от пуша
2. Требуем обязательный апрув от одного ревьюера для мержа ПР

ДЗ. Инструкция. 5. Настройка правила



Создаем шаблон для пулл реквестов

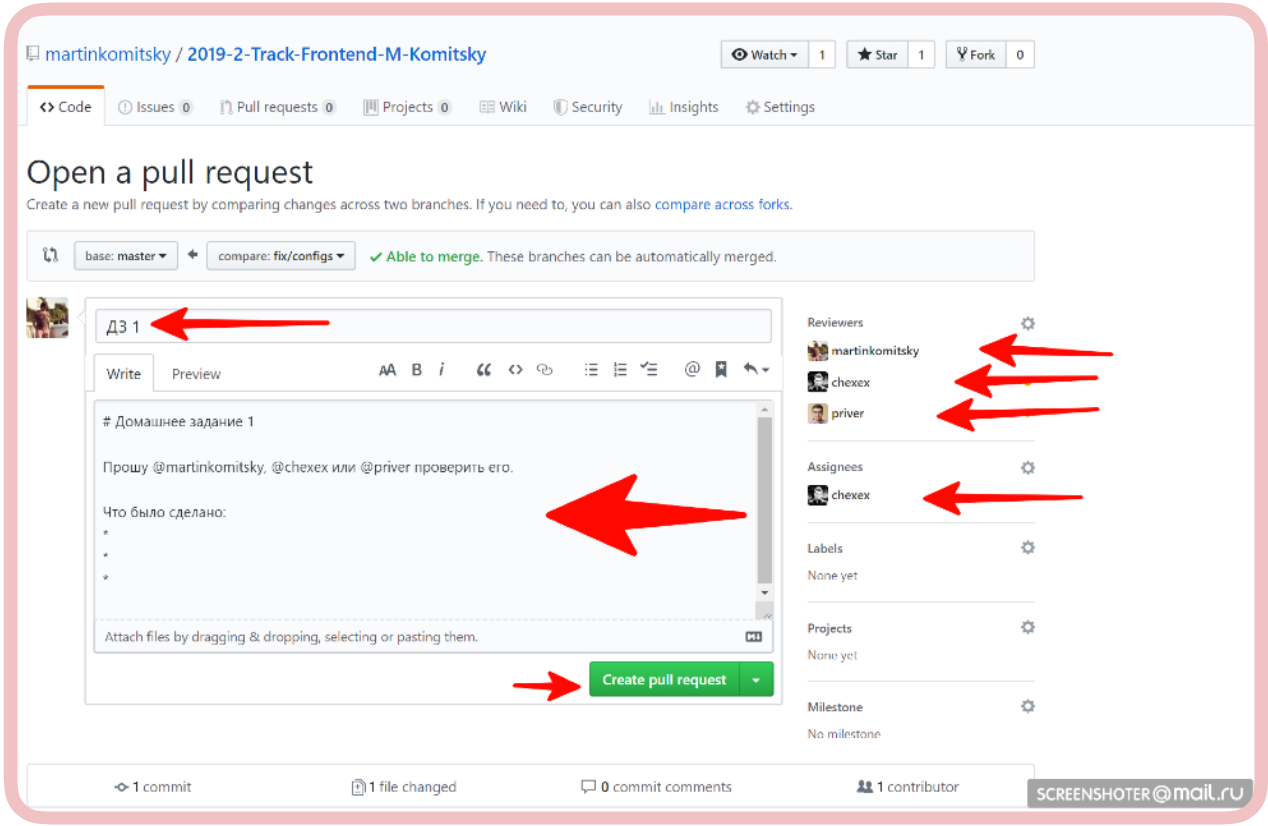
1. Создаем файл с именем **pull_request_template.md** в корне проекта
2. Содержимое должно быть следующим:

    1. # Домашнее задание №
    2. Прошу @martinkomitsky, @chexex или @priver проверить его.
    3. Что было сделано:
    4. *
    5. *

Сдаем ДЗ на проверку

1. После выполнения домашнего задания, создаем пр в ветку `master`
2. Добавляем martinkomitsky, chexex и priver в поле reviewers
3. Добавляем того, кто выдал домашнее задание (лектора конкретной лекции) в поле `assignee`
4. В теме **обязательно** пишем номер ДЗ, в описании опционально пишем то, что сделано

ДЗ. Инструкция. 7. Сдача ДЗ



Ожидаем проверки и вносим правки

1. Жмем на большую зеленую кнопку и ждем комментариев
2. Если все выполнено корректно - выполняется мерж
3. Если есть недочеты - будут оставлены замечания, которые надо исправить и запушить в текущий PR
4. Мы видим правки, повторяем процесс ревью
5. Пересоздавать PR не нужно

Сдача ДЗ

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

ДЗ. Критерий оценки

1. Для того, чтобы успешно сдать домашнее задание без применения штрафов (снятие 40% баллов) - нужно вовремя заблаговременно создать PR
2. Срок сдачи каждого задания - 2 недели с момента его выдачи на лекции, но не позднее 23:59:59 дня перед лекцией
3. Если вы уложились с решением ДЗ в 2 недели, то получаете к максимальным 10 баллам еще дополнительных 2 за "скорость"
4. Если вы не уложились в 2 недели, но сдали ДЗ на семинаре, то вы получаете максимально 10 баллов
5. Если вы не сдали ДЗ на семинаре текущего модуля, то применяется штраф 40% от максимальных 10 баллов, что делает максимумальную оценку после штрафа равной 6
6. Максимальной оценкой оценивается решение, которое было выполнено правильно с первого раза
7. Если были допущены сильные недочеты, то баллы снижаются в зависимости от тяжести положения

Просьба

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

GH-Pages

1. Зайти на https://travis-ci.org
2. Подключить к своему github
3. Перейти в travis в настройки своего репозитория
4. Добавить в Environment Variables repo_token и токен
5. Зайти на https://github.com/settings/tokens/new
6. В строке Token description указать "repo_token"
7. В Select scopes выбрать "repo"
8. В package.json задать ключ "homepage" (см.далее)
9. В package.json в "scripts" обновить ключ "test" (см.далее)

Бонус.

Token description

repo_token

What's this token for?

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ **repo**	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations

Бонус.

```
// package.json

{
  "name": "cra-caddy-docker",
  "homepage": "https://chexex.github.io/cra-caddy-docker/",
  ...
  "scripts": {
    ...
    "test": "CI=true react-scripts test --env=jsdom",
    "test:coverage": "npm test -- --coverage",
     ...
  },
   ...
}
```

Полезные ссылки

Читать:

YDKJS: Getting Started

Смотреть:

The Wierd History Of JavaScript

JavaScript: How It's Made

Спасибо за внимание!