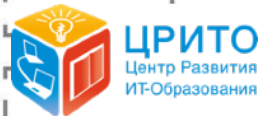
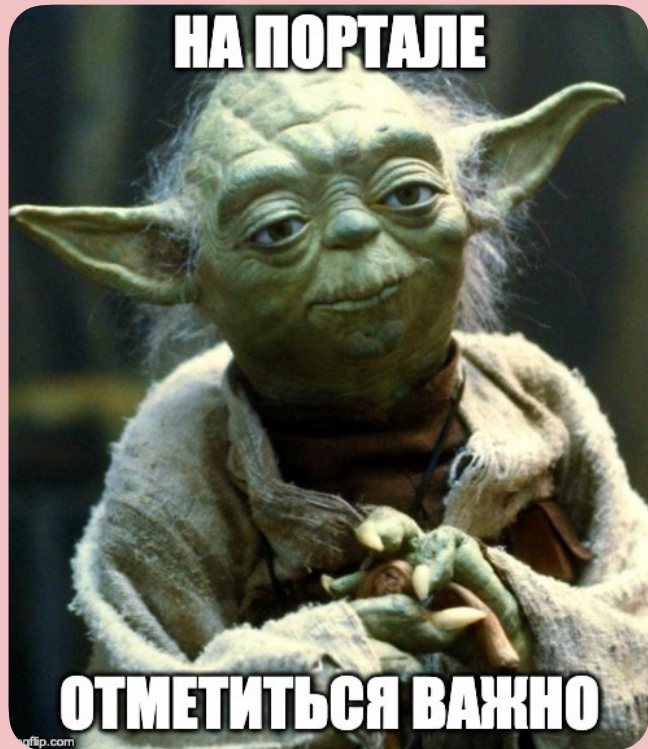




Лекция 1.

Не введение

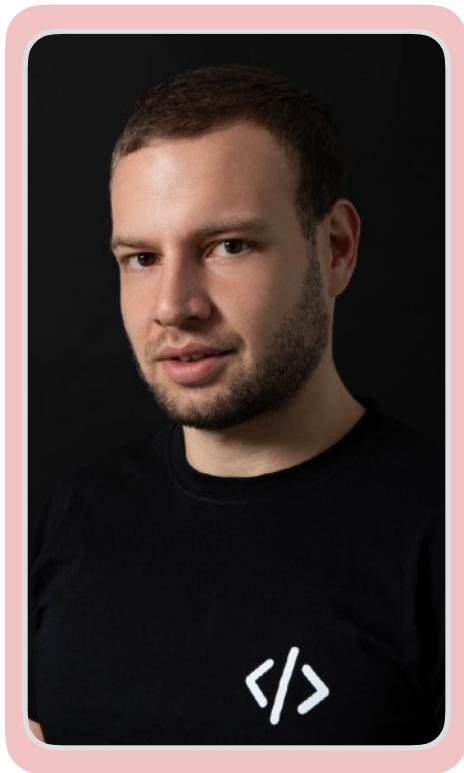




1. Отметиться на портале
2. Заполнить анкету
3. Оставить обратную связь

 **Класс!**

Это я



Алексей
Опалев
@a.opalev
Медиапроекты@Mail.RU



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.



Фронтенд разработка

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



nodejs

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



npm

Пакетный менеджер для nodejs модулей.



Линтеры

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

Основные: eslint, stylelint



CI/CD

Continious integration/continious delivery

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



Docker

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



Travis

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

Аналоги: jenkins, gitlab-ci, circleci



Генератор проектов

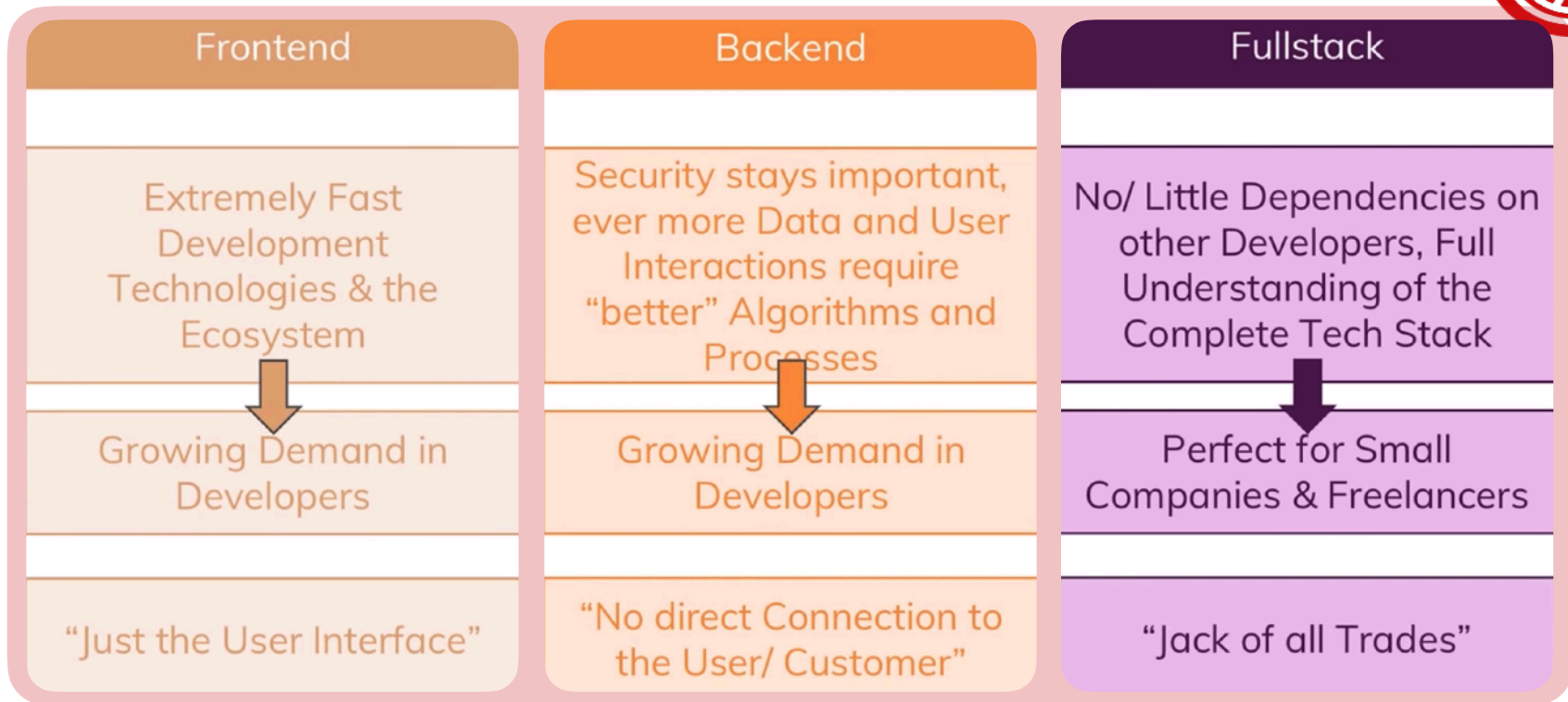
Программа/инструмент для генерации проекта.

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, ramda)
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





Что такое

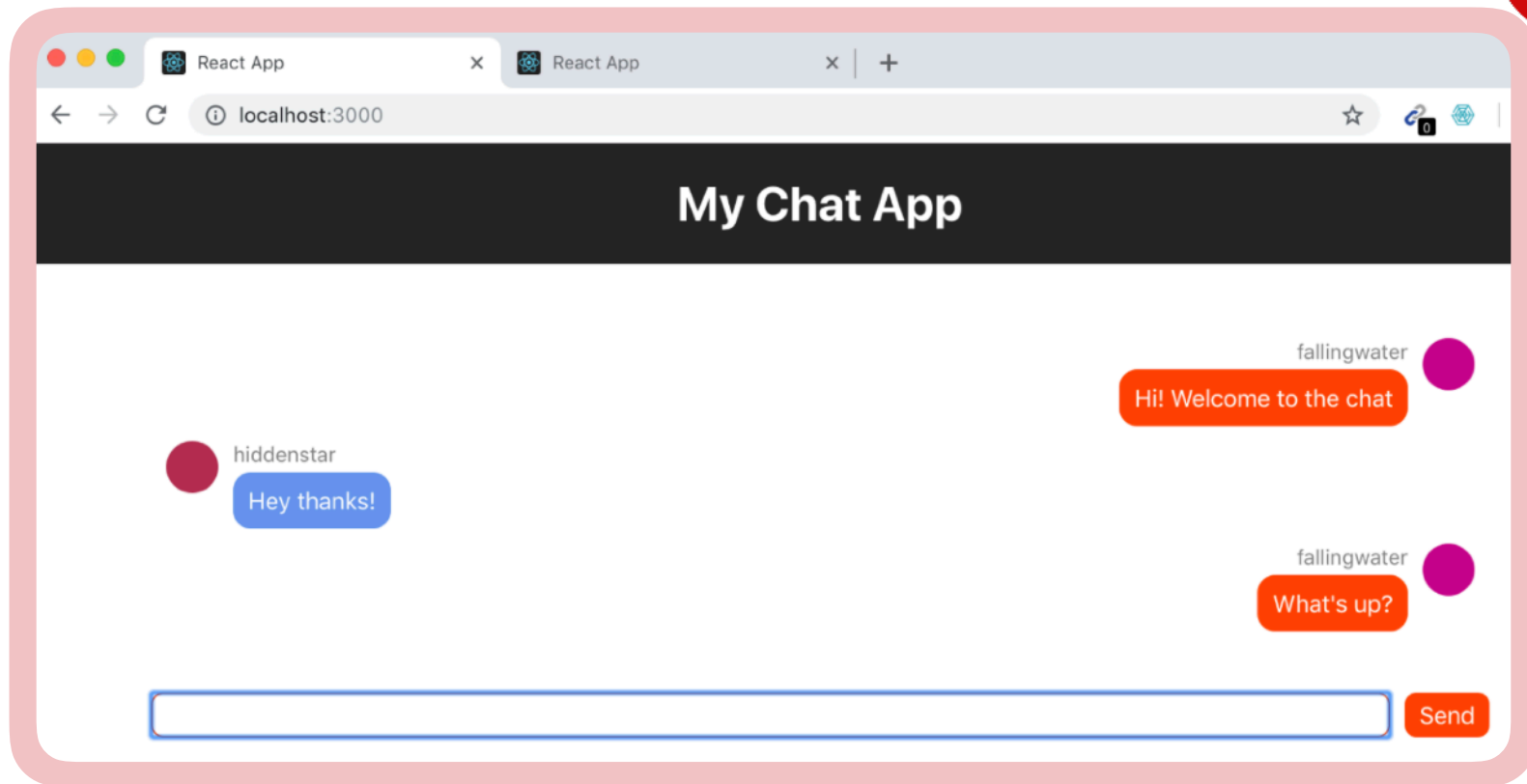
1. devtools
2. javascript
3. async
4. transport
5. I/O
6. React/Angular/Vue
7. jQuery



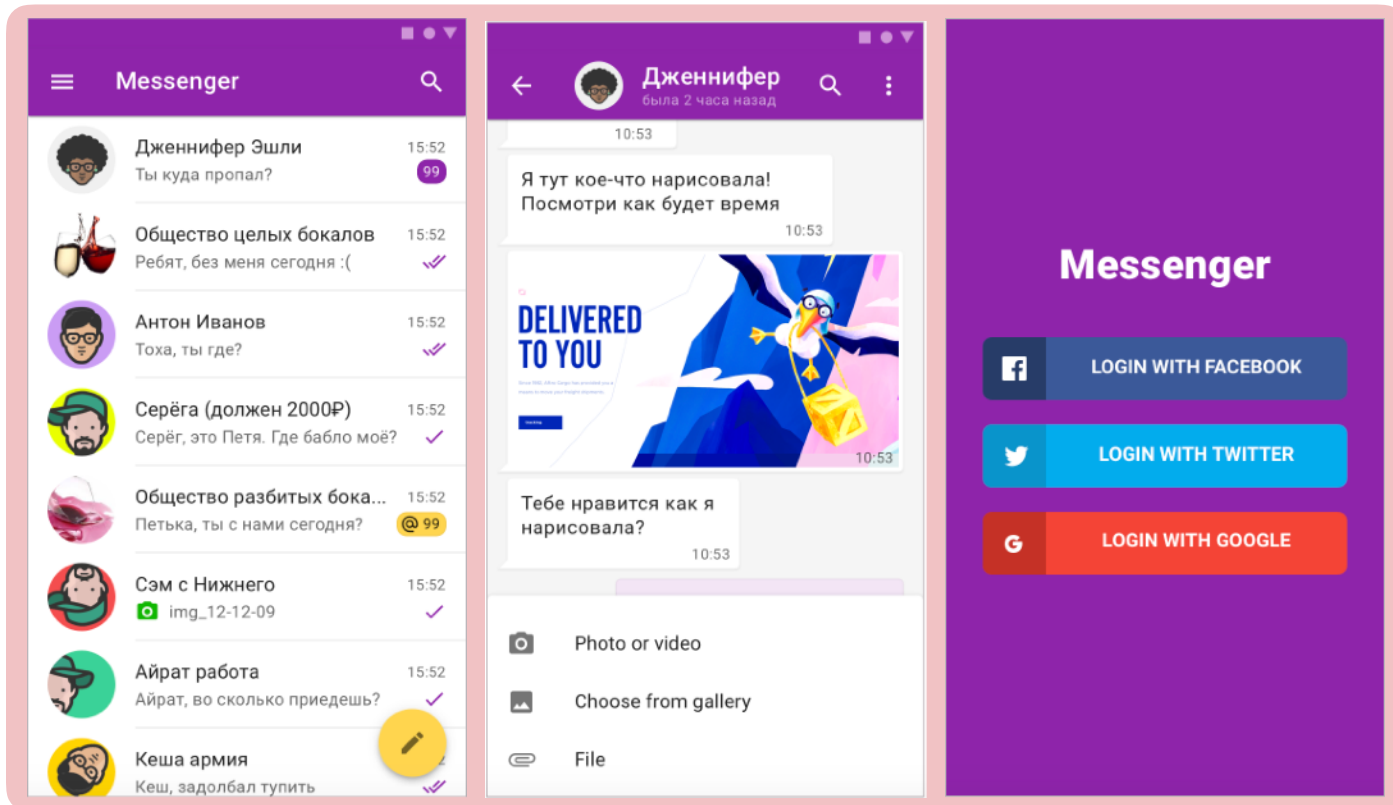
Как происходит

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

Приложение – Чат



Приложение – Чат





Оптимальный набор задач

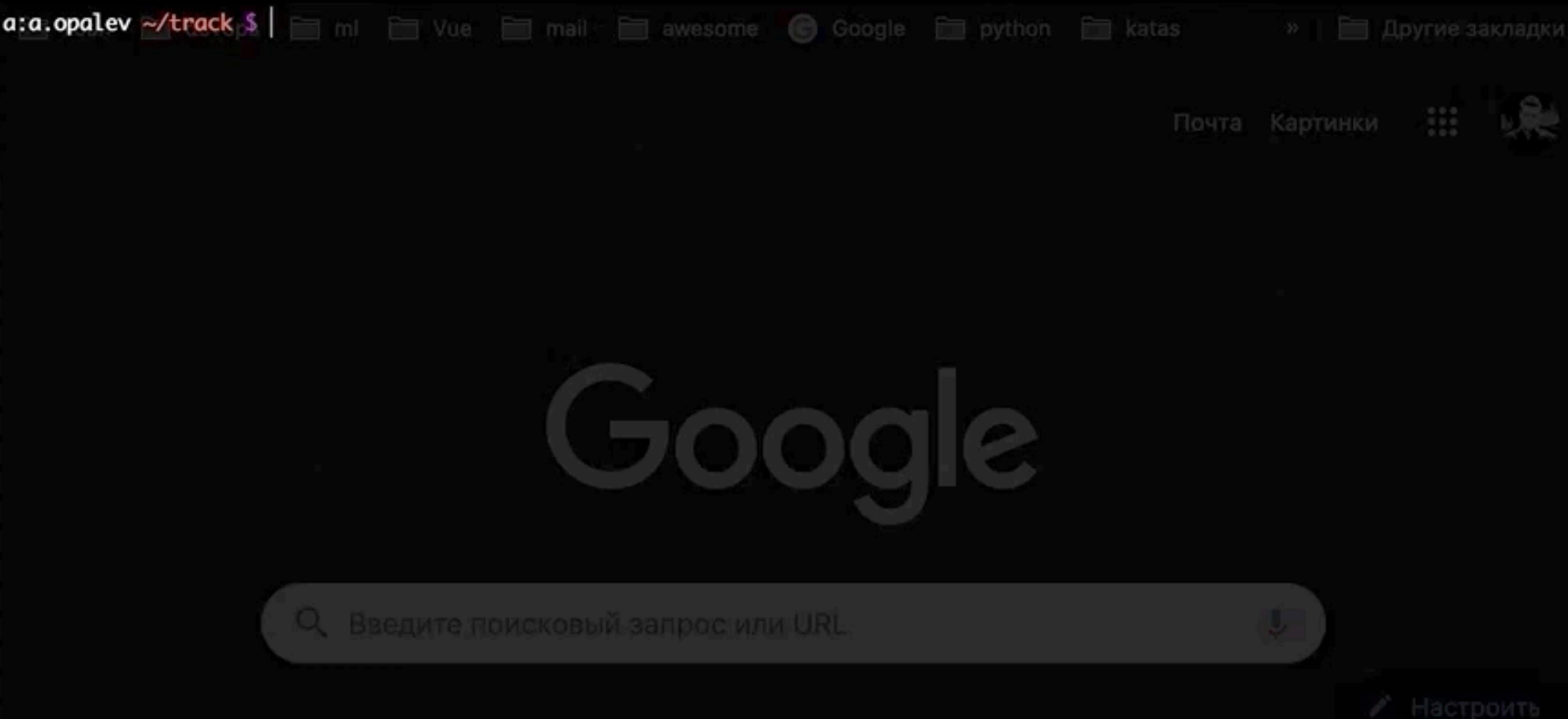
- Много верстки
- Очень много событий, которые надо обработать
- Легко эволюционирует из простого приложения в сложное



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!

ml

Vue

mail

awesome

Google

python

katas

»

Другие закладки

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

Source view

Diff to previous

History

Edit

Bl

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`.

a:a.opalev ~/track/proj \$

```
    elif in_gallery:
    49
    Note that the development build is not optimized.
    To create a production build, use npm run build.
    52         gallery.append(processed_block)
    53     else:
    54         in_gallery = False
    55         _add_pseudo_gallery(node, gallery)
    56     elif child.get('type') == 'Image':
    57         in_gallery = True
    58         gallery = [processed_block]
    59     else:
    60         if processed_block:
    61             self._build_node(processed_block, parent=node)
    62
    63     if in_list:
    64         _add_embed_list(node, embed_list, items)
```

Что происходит [open]



Compiled successfully!

ml Vue mail awesome Google python katas » Другие закладки

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

Source view Diff to previous History Edit

Local: http://localhost:3000/

On Your Network: http://192.168.1.67:3000/embed_list(node, embed_list, items)

49 elif in_gallery:

Note that the development build is not optimized. type == 'image' and processed_block:

To create a production build, use **npm run build**. add_block.pop('children', None)

52 gallery.append(processed_block)

53 else:

a:a.opalev ~/track/proj \$ docker-compose up --dry-run = False

Creating network "proj_webnet" with the default driver gallery(node, gallery)

Creating proj_frontend_1 ... done if child.get('type') == 'image':

a:a.opalev ~/track/proj \$ | in_gallery = True

58 gallery = [processed_block]

59 else:

60 if processed_block:

61 self.build_node(processed_block, parent=node)

62

63 if in_list:

64 add_embed_list(node, embed_list, items)

65 elif in_gallery:



От сложного

К сложному. Но чуть-чуть проще.



Веб приложение

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

Структура проекта



```
proj ~/track/proj
├── coverage
├── docker
├── node_modules library root
├── public
├── src
├── .babelrc
├── .dockerignore
├── .editorconfig
├── .env
├── .eslintignore
├── .eslintrc.json
├── .gitattributes
├── .gitignore
├── .nvmrc
├── .prettierrc
├── .travis.yml
├── .yo-rc.json
├── docker-compose.yml
├── docker-compose-prod.yml
├── jest.config.js
├── package.json
├── package-lock.json
├── README.md
└── stylelint.config.js
```

```
src
├── actions
│   └── counter.js
├── assets
├── components
│   ├── Counter.js
│   └── Header.js
├── constants
│   └── ActionTypes.js
├── containers
│   └── CounterContainer.js
├── reducers
│   ├── counter.js
│   └── index.js
├── routes
│   └── index.js
├── store
│   ├── index.js
│   ├── storeDev.js
│   └── storeProd.js
├── styles
├── tests
├── utils
└── index.js
```



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
```




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
```



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`



Как сдавать дз

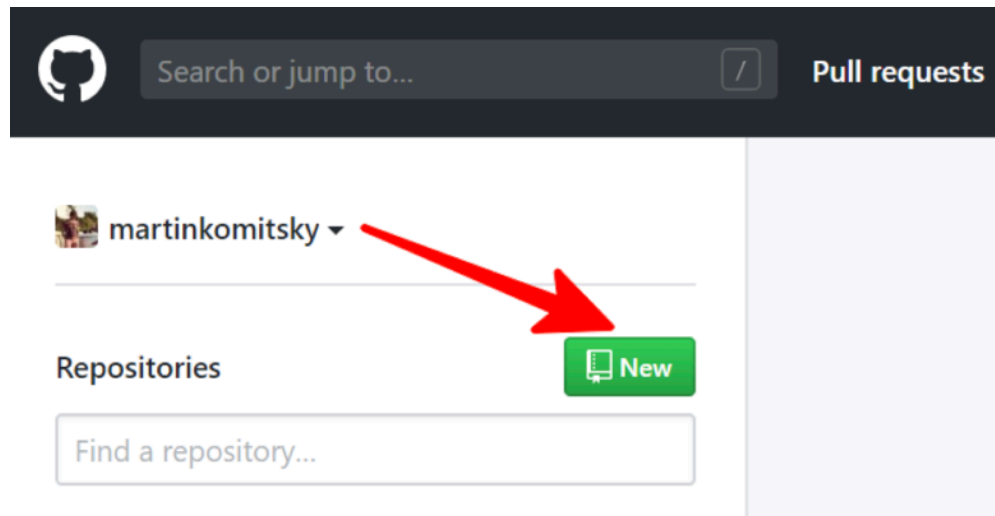
1. Выполнить локально задачи
2. Закоммитить изменения в ветке ``devel``
3. Запустить ветку в свой гитхаб
4. Поставить ПР с ``devel`` на ``master``
5. Запросить ревью ПР у меня [<https://github.com/chexex>]

Д3. Инструкция. 1. Новый репозиторий



Создаем репозиторий

1. Заходим на [GitHub](#)
2. Создаем новый репозиторий





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

Д3. Инструкция. 2. Правильное имя.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner

 martinkomitsky ▾

Repository name *

2019-2-Track-Frontend-M-Komitsky ✓

Great repository names are short and memorable. Need inspiration? How about **potential-palm-tree**?

Description (optional)

Учебный проект



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

SCREENSHOTER@mail.ru



Добавить преподавателей

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

Д3. Инструкция. 3. Коллабораторы



martinkomitsky / 2019-2-Track-Frontend-M-Komitsky

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Options

Collaborators

Webhooks

Notifications

Integrations & services

Deploy keys

Moderation

Interaction limits

Collaborators Push access to the repository

Alexey Opalev
Awaiting chexex's response

Copy invite link Cancel invite

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

priver Add collaborator

SCREENSHOTER@mail.ru



Создание правила для веток

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

Д3. Инструкция. 4. Создание правила



martinkomitsky / 2019-2-Track-Frontend-M-Komitsky

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Moderation

Interaction limits

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

The default branch is set to `master`. To change this setting, [add another branch](#).

Branch protection rules

[Add rule](#)

Define branch protection rules to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to branch protection rules? [Learn more](#).

No branch protection rules defined yet.

SCREENSHOTER@mail.ru



Настройка правила для веток

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

Д3. Инструкция. 5. Настройка правила



martinkomitsky / 2019-2-Track-Frontend-M-Komitsky

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Security Insights Settings

Options
Collaborators
Branches
Webhooks
Notifications
Integrations & services
Deploy keys
Moderation
Interaction limits

Branch protection rule

Branch name pattern
master

Rule settings

Protect matching branches
Disables force-pushes to all matching branches and prevents them from being deleted.

- ☒ **Require pull request reviews before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.
Required approving reviews: 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 signed commits**
Commits pushed to matching branches must have verified signatures.
- ☐ **Include administrators**
Enforce all configured restrictions for administrators.

Create

SCREENSHOTER@mail.ru



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, @schexex 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



martinkomitsky / 2019-2-Track-Frontend-M-Komitsky

Watch

1

Star

1

Fork

0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

Settings

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: fix/configs

✓ Able to merge. These branches can be automatically merged.

ДЗ 1

WritePreview

AA B i « > ↺ ⋮ ⋮ ⋮ @ 📎 ↶

Домашнее задание 1

Прошу @martinkomitsky, @chexex или @priver проверить его.

Что было сделано:

*
*
*

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

Create pull request

Reviewers

martinkomitsky

chexex

priver

Assignees

chexex

Labels

None yet

Projects

None yet

Milestone

No milestone

1 commit

1 file changed

0 commit comments

1 contributor

SCREENSHOTER@mail.ru



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 запустить 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`

Д3. 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 Д3 в 2 недели, то получаете к максимальным 10 баллам еще дополнительных 2 за "скорость"
4. Если вы не уложились в 2 недели, но сдали Д3 на семинаре, то вы получаете максимально 10 баллов
5. Если вы не сдали Д3 на семинаре текущего модуля, то применяется штраф 40% от максимальных 10 баллов, что делает максимальную 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 вогнуть 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)



Token description

repo_token

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

- | | |
|---|--------------------------------------|
| <input checked="" type="checkbox"/> repo | Full control of private repositories |
| <input checked="" type="checkbox"/> repo:status | Access commit status |
| <input checked="" type="checkbox"/> repo_deployment | Access deployment status |
| <input checked="" type="checkbox"/> public_repo | Access public repositories |
| <input checked="" type="checkbox"/> repo:invite | Access repository invitations |

Бонус.



```
1. // package.json
2. {
3.   "name": "cra-caddy-docker",
4.   "homepage": "https://chexex.github.io/cra-caddy-docker/",
5.   ...
6.   "scripts": {
7.     ...
8.     "test": "CI=true react-scripts test --env=jsdom",
9.     "test:coverage": "npm test -- --coverage",
10.    ...
11.  },
12.  ...
13.}
```




Полезные ссылки

Читать:

[YDKJS: Getting Started](#)

Смотреть:

[The Wierd History Of JavaScript](#)
[JavaScript: How It's Made](#)



Спасибо за внимание!