

Лекция 11

Оптимизация, Deploy, безопасность

Дмитрий Зайцев
Мартин Комитски



План на сегодня

1. Оптимизация

- Зачем?
- Что можно оптимизировать?
- Document
- Сетевое взаимодействие
- Вычисления
- Прочее
- Практика?

2. Автоматизация

- Линтеры
- Хуки
- Контейнеры

3. Безопасность

- Глоссарий
- Куки

Минутка бюрократии

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



Оптимизация

Зачем?

Оптимизация. Для чего нужно оптимизировать?

- Пользовательский опыт
 - Снижение скорости загрузки страницы до 50% сокращает конверсию
 - Снижение быстродействия страницы также увеличивает число отказов
- SEO
 - Скорость загрузки и быстродействие страниц влияет на ранжирование в поиске
- Нагрузка на сервера
 - Количество запросов и объем передаваемых данных прямо влияют на производительность сервера

Что можно
оптимизировать?

Оптимизация. Что можно оптимизировать?

- Document
 - Количество элементов
 - Сложность/специфичность селекторов
 - Периодичность пересчета/перерисовки
- Сетевое взаимодействие
 - Блокирующие ресурсы
 - Размер ресурсов
 - Количество запросов
- Вычисления
 - Сложные манипуляции с данными
 - Неоптимальное использование обработчиков событий

Document

Оптимизация. Document

Важно:

- $O(\text{Nodes} * \text{Selectors} * \text{Depth})$
- Сокращать количество элементов до минимально необходимого
- Не использовать недостаточно специфичные комбинаторы

1. `.my-class span {color: red}`
2. `.my-class > span {color: red}`
3. `.my-class_red {color: red}`
- 4.
5. `<html>`
6. `<div>`
7. `text`
8. `</div>`
9. `<div class="my-class">`
10. `red text`
11. `</div>`
12. `</html>`

Оптимизация. Document

- Избегать повторного вычисления стилей и пересчета Layout
- Триггеры пересчета Layout (reflow)
 - Изменение «геометрических свойств» элемента
 - Изменение стилей или свойств, влияющий на стили (<https://csstriggers.com/>)
 - Обращение к свойствам элемента, измененным после последнего пересчета

1. `node.classList.add('my-class');`
2. `console.log(node.offsetHeight);`

- И многое другое
- Использовать «слои», но не злоупотреблять ими
 - `will-change: value;`
 - `transform: translateZ(0);`
- Ограничивать размеры областей перерисовки и ее сложность

Оптимизация. Document

- BEM

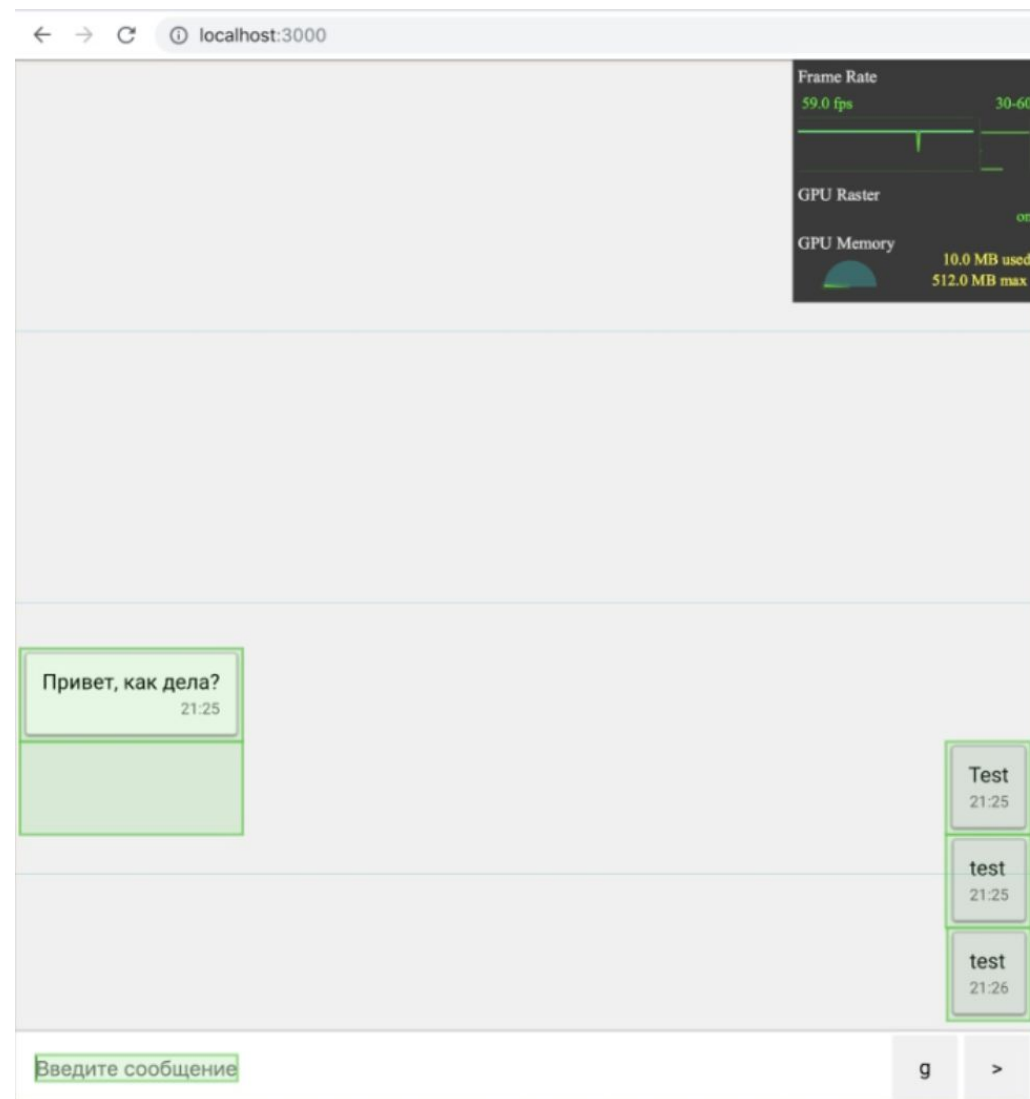
1. `<form class="search-form">`
2. `<div class="search-form__content">`
3. `<input class="search-form__input">`
4. `<button class="search-form__button">Найти</button>`
5. `</div>`
6. `</form>`

- CSS Modules

1. `// для CRA достаточно добавить .module к имени стилевого файла`
2.
3. `import styles from './style.css';`
4. `// import { className } from './style.css';`
5. `element.innerHTML = '<div class="' + styles.className + '">';`

Оптимизация. Document

- Инструменты анализа производительности
 - [rendering settings](#)
 - [perfomance-tools](#)



Сетевое взаимодействие

Оптимизация. Сетевое взаимодействие

- Загрузка скриптов и стилей может откладывать момент отрисовки
- Инлайн критических ресурсов
- Отложенная загрузка
 - defer, async, preload, prefetch
 - js загрузчики

Сетевое взаимодействие. Inline, async, defer

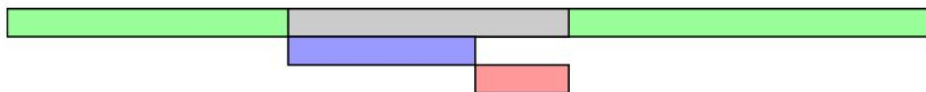
- [async-vs-defer-attributes](#)
- <https://stackoverflow.com/a/39711009>

Legend

- HTML parsing
- HTML parsing paused
- Script download
- Script execution

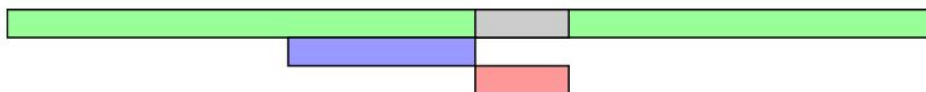
<script>

Let's start by defining what **<script>** without any attributes does. The HTML file will be parsed until the script file is hit, at that point parsing will stop and a request will be made to fetch the file (if it's external). The script will then be executed before parsing is resumed.



<script async>

async downloads the file during HTML parsing and will pause the HTML parser to execute it when it has finished downloading.



<script defer>

defer downloads the file during HTML parsing and will only execute it after the parser has completed. **defer** scripts are also guaranteed to execute in the order that they appear in the document.



Сетевое взаимодействие

- JS-бандлы

```
1.  /src  
2.  └─ main.js  
3.  └─ component.js  
4.
```



```
5.  /dist  
6.  └─ bundle.js  
7.
```

Сетевое взаимодействие

- Caching (expires, cache-control, max-age)
- Gzip (Content-Encoding)
- CDN
- Load Balancers
- Разбиение кода
- Tree Shaking/Dead code elimination
- Memoization
- Service Workers
- Progressive page loading (events, lazy images and iframes)
- Code Minification/Uglification
- Image, font optimization
- AMP/Instant Articles/Telegram's Instant View/Yandex Turbo
- [Resource hints](#)

Сетевое взаимодействие

- CSS-спрайты

```
1. .foo {  
2.     background: url('../assets/gift.png?sprite');  
3. }
```



```
4. .foo {  
5.     background: url(/sprite.png?[hash]) no-repeat;  
6.     background-position: -100px -0px;  
7. }  
8.
```



Сетевое взаимодействие

- SVG-спрайты

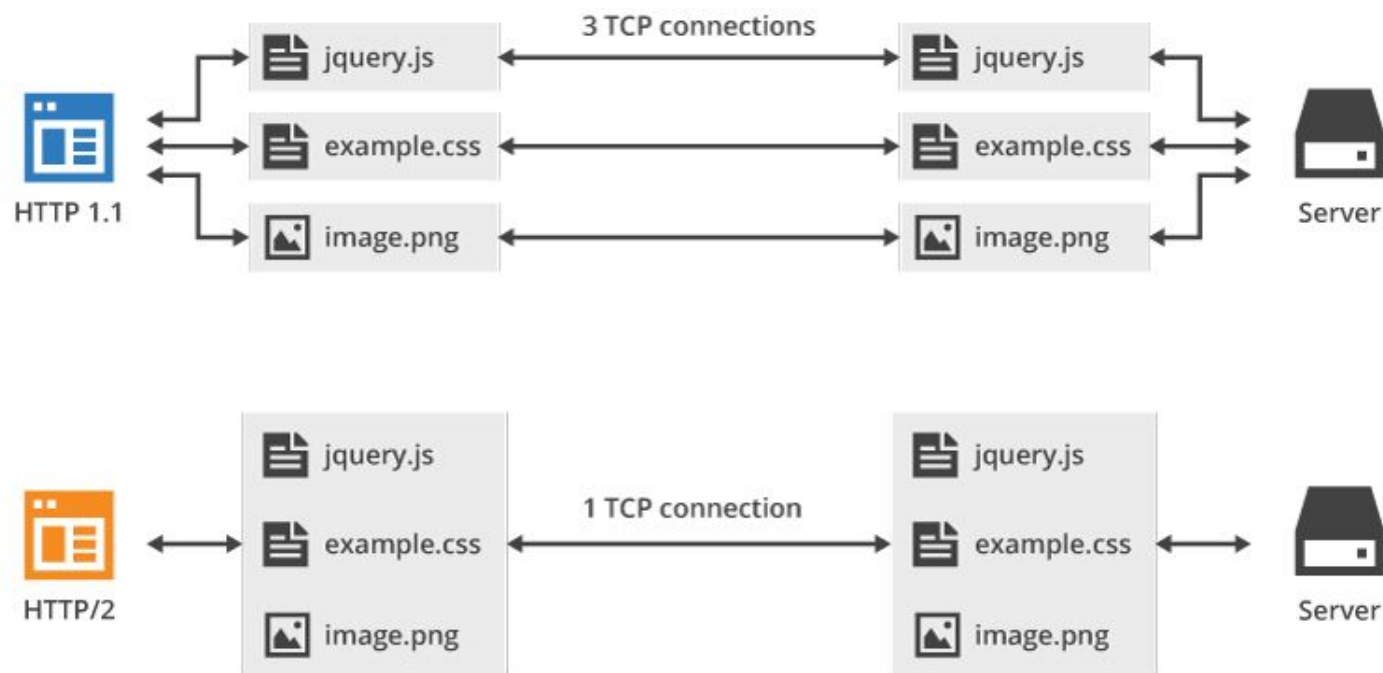
- [Пример](#)

```
1.  <svg version="1.1"
2.    xmlns="http://www.w3.org/2000/svg"
3.    xmlns:xlink="http://www.w3.org/1999/xlink"
4.  >
5.    <symbol id="first" viewBox="0 0 64 64">...</symbol>
6.    <symbol id="second" viewBox="0 0 64 64">...</symbol>
7.  </svg>
8.
9.  <svg>
10.    <use xlink:href="#first"></use>
11.  </svg>
12.
13.  <svg>
14.    <use xlink:href="#second"></use>
15.  </svg>
16.
17.
```

Сетевое взаимодействие

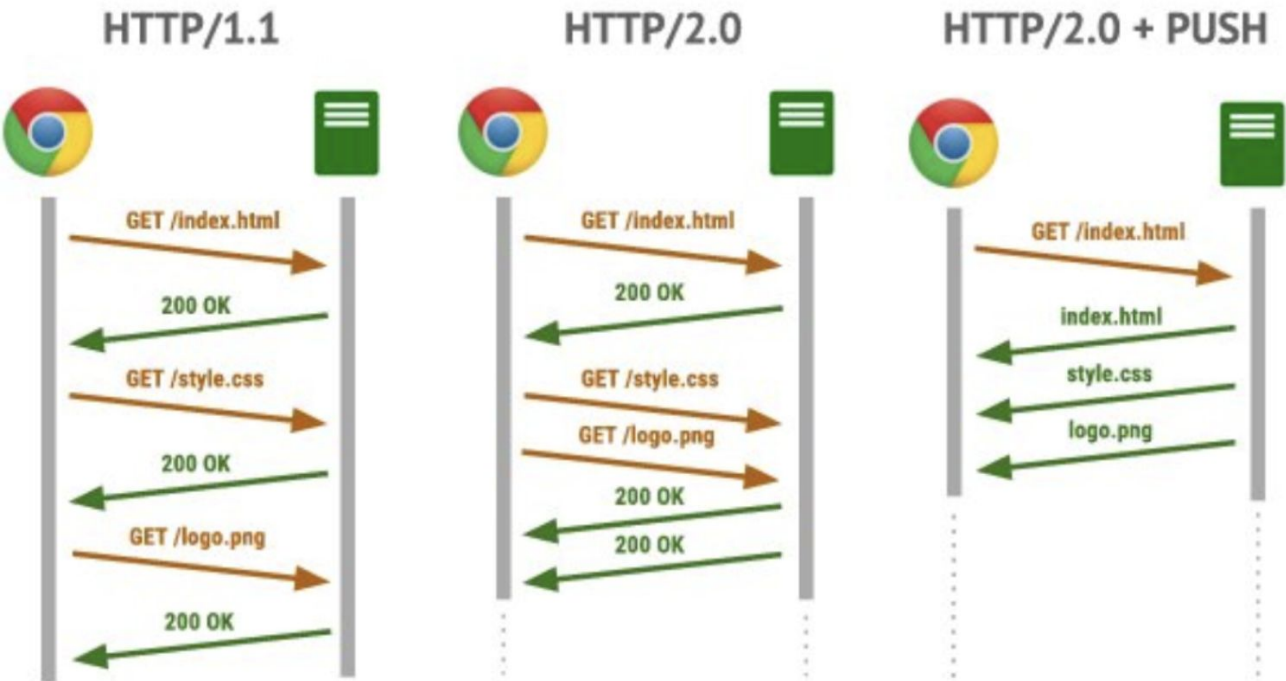
- HTTP/2
- одно соединение для нескольких запросов
- в отличие от Keep-alive ответы могут приходить одновременно

Multiplexing



Сетевое взаимодействие

- [HTTP/2 Server Push](#)
 - позволяет превентивно отправлять клиенту необходимые ресурсы
 - немного ломает логику кеширования на клиенте



HTTP/1.1	
Request HTML	/index.html
Request content	style.css
Request content	logic.js
Request content	image.jpg

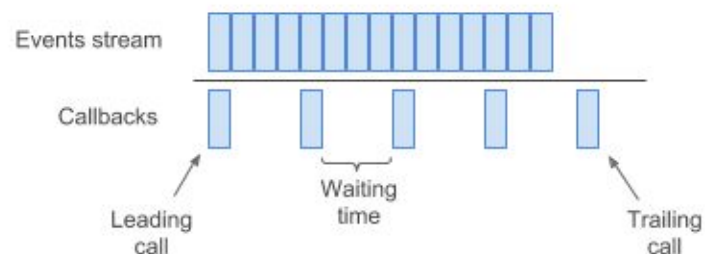
HTTP/2 Without Server Push	
Request HTML	/index.html
Request content	style.css
Request content	logic.js
Request content	image.jpg

HTTP/2 With Server Push	
Request HTML	/index.html
Request content	style.css
Request content	logic.js
Request content	image.jpg

Вычисления

Вычисления

- throttling – вызов функции не чаще чем один раз за определенный интервал времени



- debouncing – вызов функции один раз, для серии вызовов происходящих чаще, чем заданный интервал времени



Прочее

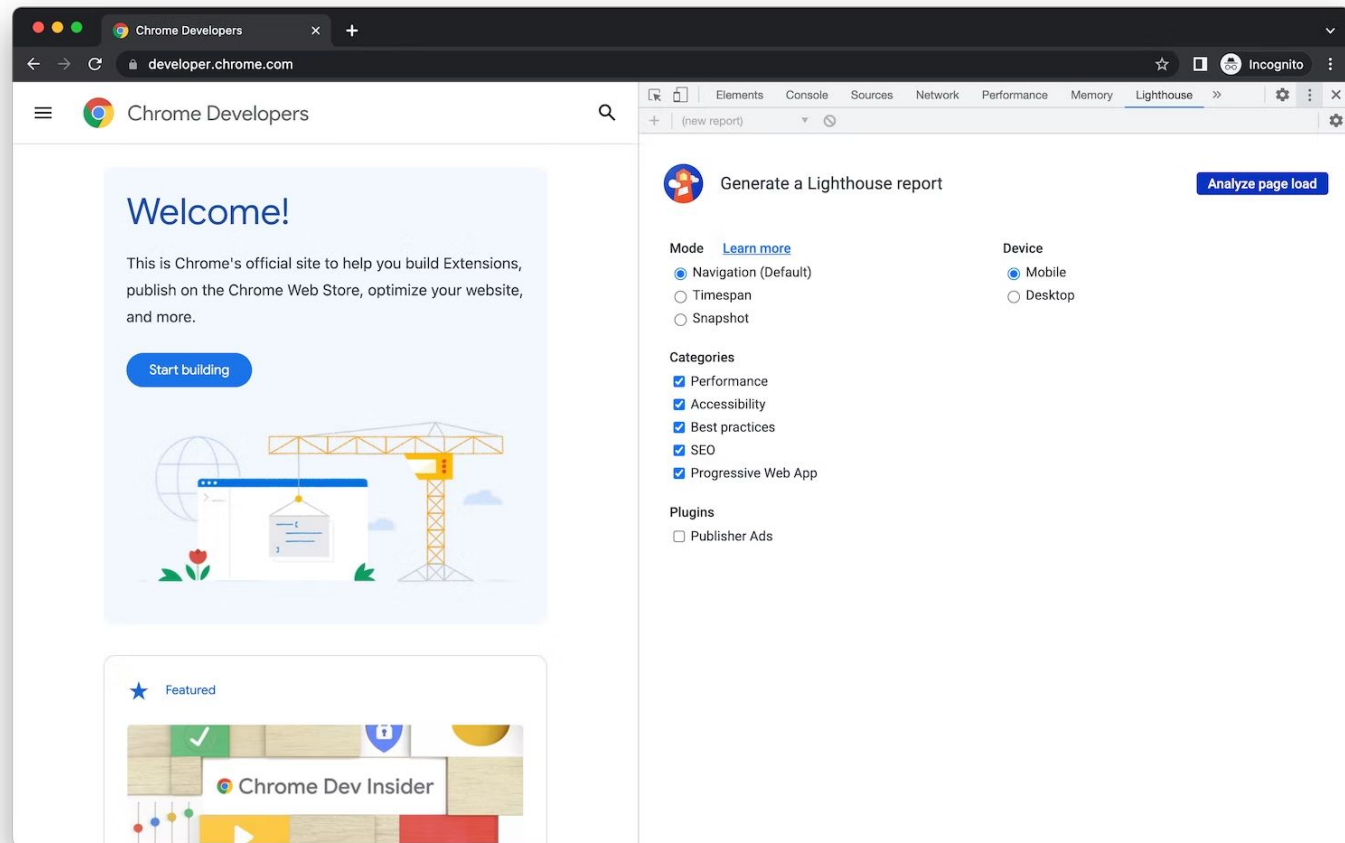
Прочее

Дополнительные пути для оптимизации:

- Add long-term headers expiration dates
- Make fewer HTTP requests
- Avoid URL redirect
- Avoid empty SRC or HREF
- Remove duplicate JavaScript and CSS
- Make AJAX cacheable
- Avoid HTTP 404 (Not Found) error
- Remove unsupported components
- Use cookie-free domains
- Reduce cookie size
- Remove unnecessary CSS rules
- Don't scale images in HTML
- Reference images in the HTML
- Reduce DNS lookups
- Resize images

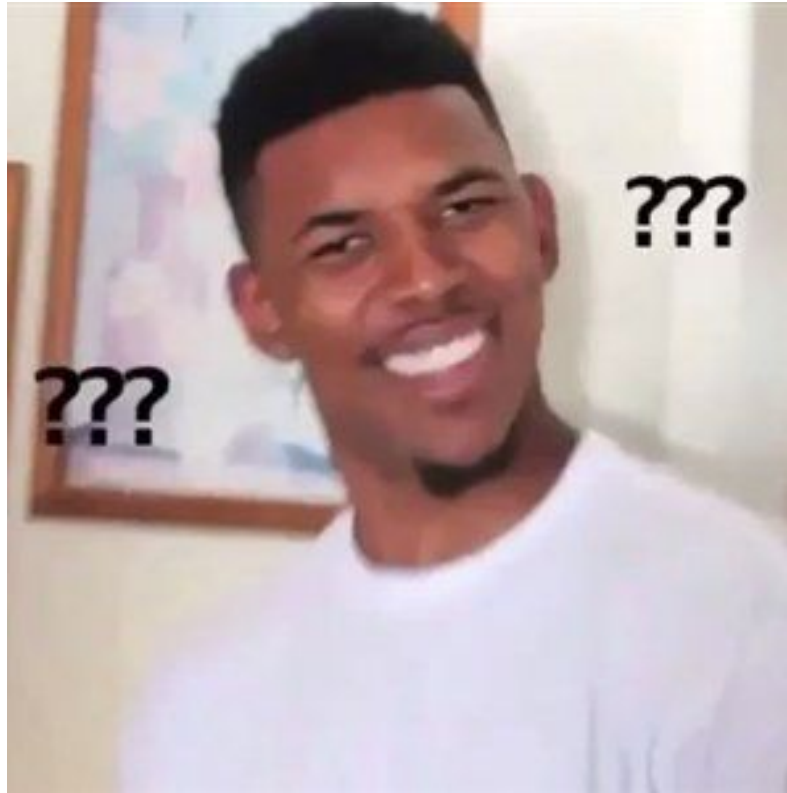
Прочее. Инструмент

- [Lighthouse](#)



Оптимизация?

Вопросы?



Практика

Практика

- Eject
- Custom loader
- Performance
- Render
- Lighthouse

Полезные ссылки

- <https://habr.com/ru/company/badoo/blog/320558/>
- <https://www.udacity.com/course/browser-rendering-optimization--ud860>
- <https://habr.com/ru/post/316618/>
- <https://tproger.ru/translations/how-to-boost-frontend/>
- <https://www.smashingmagazine.com/2020/01/front-end-performance-checklist-2020-pdf-pages/>
- <https://www.imperva.com/learn/performance/front-end-optimization-feo/>
- <https://techbeacon.com/app-dev-testing/23-front-end-performance-rules-web-applications>
- <https://mentormate.com/blog/front-end-optimization-habits-effective-developers/>
- <https://habr.com/ru/company/yandex/blog/195198/>



Перерыв! (10 минут)

Препоd (с)

Deploy,
безопасность

Вопросы с собеса

- Как попасть в ваше приложение по сети
- Как код попадает в продакшн
- Что такое автоматизация
- Что происходит, после того как вы запустили код в мастер
- Кто должен отвечать за деплой

Вопросы с собеседа (специализированные)

- Сколько веток нужно для разработки
- Зачем разделять дев и прод окружения
- Что такое веб сервер
- Что такое DNS
- Что такое "сборка в облаке"
- Что такое ci/cd
- Что такое докер
- Что такое линтер

Автоматизация



Линтеры

Линтер – анализатор кода. Проверяет код на стилистические, синтаксические и специфичные для языка ошибки.

Зачем использовать?

- Повышение качества ПО
- Улучшение читаемости кода
- Сокращение времени на ревью

Когда использовать?

- Во время написания кода (IDE)
- Перед коммитом (гит хуки)
- При сборке приложения (после пуша)

GIT HOOKS

Хуки - команды/скрипты, которые будут выполнены до или после git команды (commit, push, pull, etc)

```
mv .git/hooks/pre-commit.sample .git/hooks/pre-commit
$ cat > .git/hooks/pre-commit << EOF
> echo 'OMG COOMIT IS DONE!'
> EOF
```

Дополнительно:

<https://githooks.com>

HUSKY

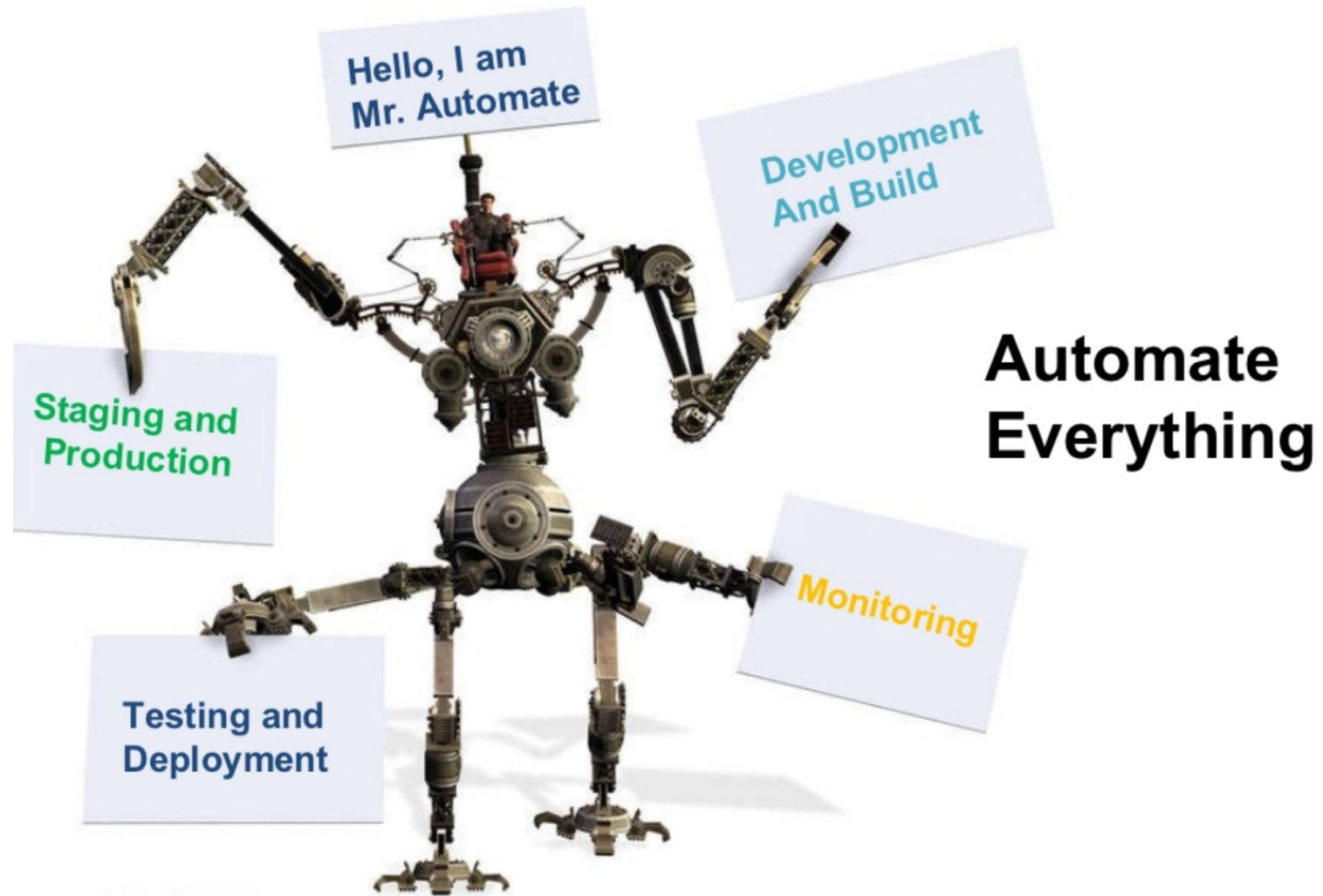
Позволяет описывать git хуки из package.json

```
$ npm i --saveDev husky lint-staged prettier
```


HUSKY

```
"husky": {  
  "hooks": {  
    "pre-commit": "lint-staged"  
  }  
},  
"lint-staged": {  
  "src/**/*.{js,jsx,ts,tsx,json}": [  
    "prettier --write",  
    "git add"  
  ]  
},
```

CI/CD



Непрерывная интеграция

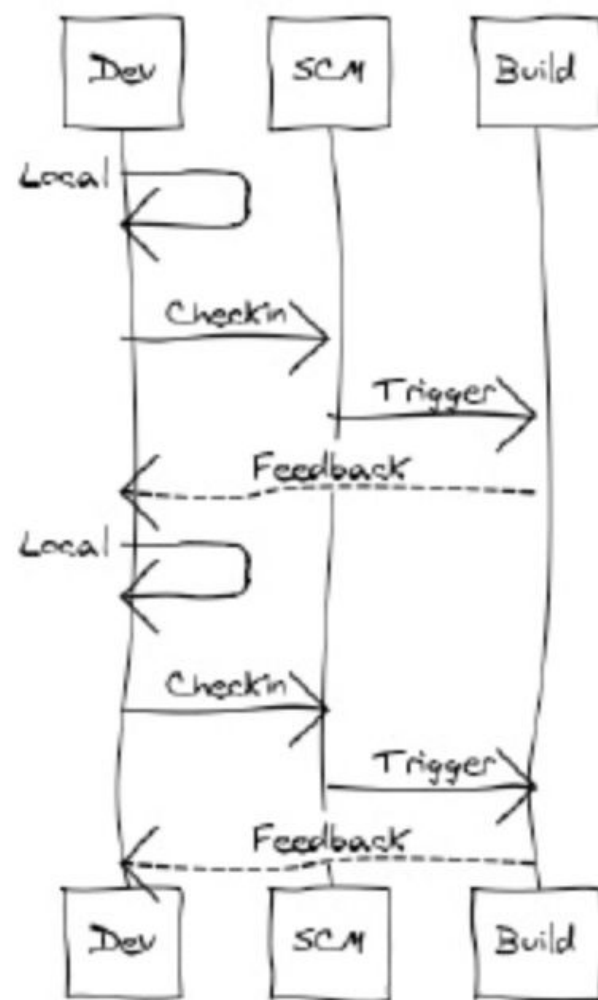
CI (Continuous Integration) - практика слияния выполненных разработчиками работ в основное хранилище/репозиторий (github/gitlab/bitbucket) – trunk/mainline. Непрерывно.

Непрерывная доставка

CD[E] (Continuous Delivery) - практика автоматизации всего процесса релиза ПО. Выполняется CI + подготовка приложения к выпуску на боевые сервера. Гарантируется высокое качество поставляемого ПО для возможности совершить релиз любое время.

Непрерывное развертывание

CD (Continuous Deployment) - выполняется CDE + автоматический деплой в продакшн с перезапуском сервером приложения при необходимости.



CDE & CD

CONTINUOUS DELIVERY



CONTINUOUS DEPLOYMENT



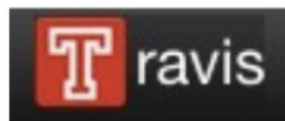
Сервисы



Jenkins



CruiseControl.rb
Continuous Integration for Ruby



pm_{ase}



Docker

Docker – проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, способных выполняться в любой среде.

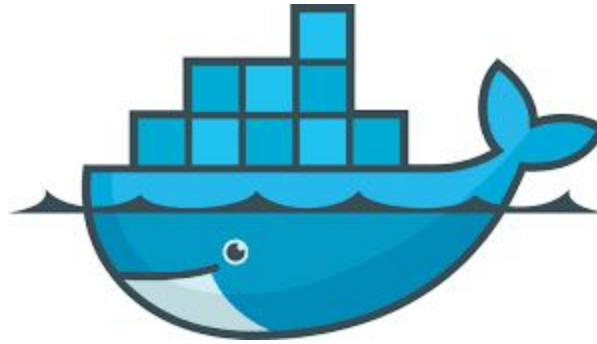
Цель





- Изоляция окружения
- Ограничение ресурсов
- Упрощение дистрибьюции

Дополнительно:

<https://docs.docker.com/get-started/>

Docker



-  Каждый компонент системы в отдельном контейнере
-  Контейнеры содержат в себе всю конфигурацию
-  Образы хранятся в registry
-  Образы версионятся

NGINX

NGINX – один из самых известных веб серверов. Способен выдерживать высокие нагрузки и реализовать архитектуру любой сложности. Огромный набор настроек – это плюс и минус nginx.

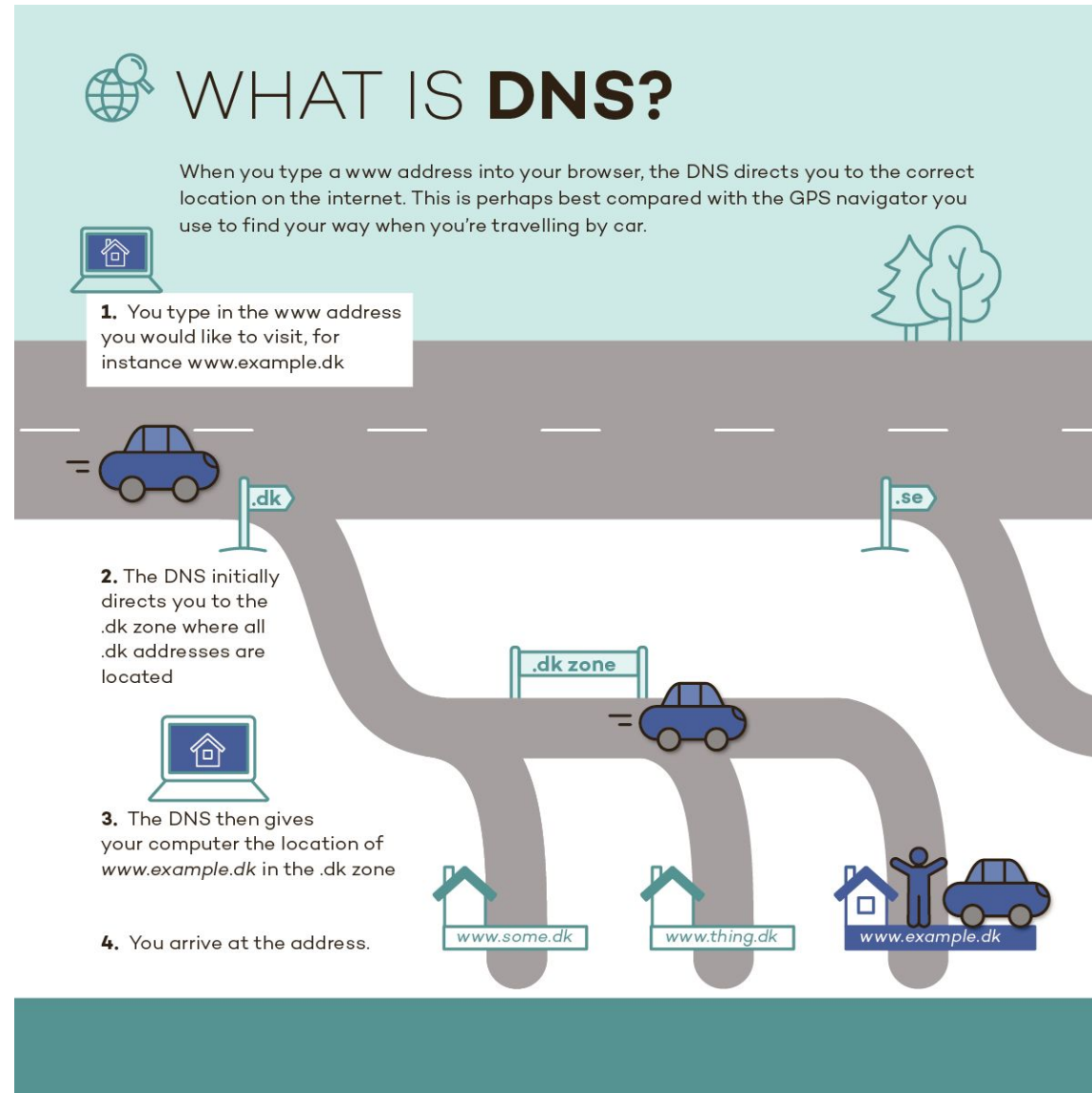
Основные кейсы для использования

- proxy
- reverse-proxy

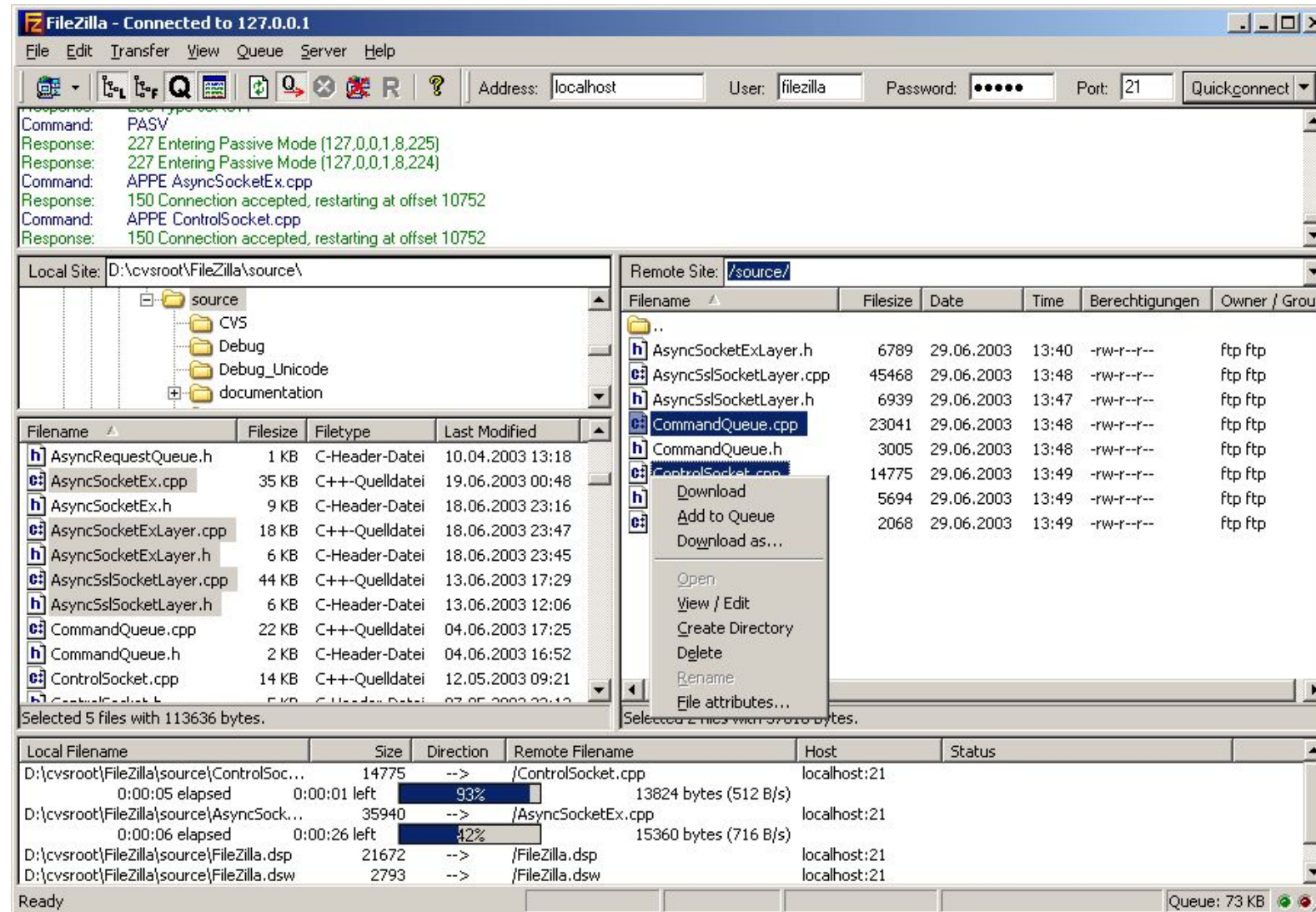
Caddy

Caddy – популярный веб сервер с поддержкой https и http2 из коробки.
Очень простой в настройке.

DNS



Long time ago...



NGINX+DOCKER+CRA

```
$ docker-compose build frontend
```

```
$ docker-compose up -d frontend
```

Dockerfile

```
# Stage 0, "build-stage" to build and compile the frontend
from node:11-alpine as build-stage
WORKDIR /app
COPY . ./
RUN npm i
RUN yarn install
RUN npm run build

# Stage 1, to have only the compiled app, ready for production
with Nginx
from nginx:1.15-alpine
COPY --from=build-stage /app/build /usr/share/nginx/html
COPY ./nginx.conf /etc/nginx/conf.d/default.conf
```


Nginx

```
server {  
    listen 80;  
    location / {  
        root /usr/share/nginx/html;  
        index index.html index.htm;  
        try_files $uri $uri/ /index.html =404;  
    }  
    include /etc/nginx/extra-conf.d/*.conf;  
}
```

docker-compose.yml

```
version: '3.6'
services:
  frontend:
    build:
      context: .
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf # to mount
        custom nginx.conf
    ports:
      - "80:80"
```

Дополнительно:

<https://docs.docker.com/compose/gettingstarted/>

Безопасность

Вопросы с собеседа

- Что такое компьютерная безопасность?
- Какие последствия могут быть, если в приложении есть риск атак? Как снизить риск атак?

Вопросы с собеса (специализированные)

- Что такое cookie
- Что такое https
- Что такое CSRF
- Что такое XSS
- Что такое CSP
- Чем авторизация отличается от аутентификации Какие знаете способы аутентификации пользователя

Компьютерная безопасность

Процесс обеспечения:

- конфиденциальности данных
- целостности данных
- доступности данных

для пользователей или клиентов информационных систем

Конфиденциальность (confidentiality)

Система обеспечивает приватное хранение личных данных пользователя.

Атаки: раскрытие информации против воли пользователя. (Disclosure attacks)

Целостность (integrity)

Система обеспечивает надежное хранение личных данных.

Атаки: изменение или уничтожение данных. (Alteration attacks)

Доступность (availability)

Система обеспечивает доступ пользователя к данным.

Атаки: отказ от обслуживания. (Denial attacks)

Терминология

Ассет/актив/ценность (Asset) - объект, представляющий интерес для злоумышленника (личные данные пользователей, вычислительные ресурсы, репутация пользователя)

Терминология

Угроза (Threat) - действие, которое ведет к потере ценности актива (изменение прав собственности, уничтожение, повреждение или раскрытие актива)

Терминология

Уязвимость (Vulnerability) - слабое место в системе (передача пользовательских данных в GET параметре)

Терминология

Риск (Risk) - наличие в системе уязвимости, и угрозы. Возможность совершения атаки.

Терминология

Атака (Attack) - реализованный риск.

Не все риски ведут к атакам, но все атаки – результат реализации риска системы.

Терминология

Ослабление угроз (Mitigation) - процесс снижения рисков в системе за счет снижения количества уязвимостей или за счет обесценивания активов.

Атаки не могут быть ослаблены. Снижать можно только риски

Cookies

Cookies – механизм хранения информации на клиенте, обеспечивая таким образом возможность идентификацию пользователя и/или его действий. Куки являются заголовком HTTP протокола.

Назначение

- Управление сеансом (логин, просмотренная лента, корзина)
- Персонализация
- Мониторинг

Дополнительно:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies> - читать обязательно

Cookies

Нужные атрибуты:

- Domain
- Expires
- Max-Age
- Path

Нужные флаги:

- Secure
- HttpOnly
- SameSite

```
Set-Cookie: id=longid; Domain=pets.mail.ru; Path=/rubrics; Expires=Wed,  
Jun 2023 07:28:00 GMT; Secure; HttpOnly; SameSite;
```

Cookies

<https://www.freecodecamp.org/news/web-security-hardening-http-cookies-be8d8d8016e1>

Аутентификация & авторизация

Аутентификация (Authentication) = логин + пароль (Кто ты?)

Процесс удостоверения, что некто действительно тот, за кого себя выдает.

Аутентификация & авторизация

Авторизация (Authorization) = доступы (permissions) (Что ты можешь?)

Набор правил, определяющих, кто какие имеет возможности

Дополнительно:

<https://stackoverflow.com/questions/6556522/authentication-versus-authorization>

HTTPS

http over tls

Дополнительно:

<https://hpbn.co/transport-layer-security-tls/>

Что такое?

- XSS
- SQL Injection
- csrf
- ssrf

Что такое?

- same origin policy
- content security policy
- hsts HTTP Strict-Transport-Security

Deploy, безопасность?

Вопросы?



Домашнее задание №11

1. Оптимизация
2. Производительность
3. Утечки
4. Linter

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

Срок сдачи

29 декабря

Спасибо за внимание!



Пока!

Присоединяйтесь к сообществам про образование в VK

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

