

ПМЗ Разработка модулей ПО.

РО 3.1 Понимать и применять принципы объектно-ориентированного и асинхронного программирования.

Тема 2. Разрабатывать модули с применением DOM API Regexp HTTP.

Лекция 17. Перенаправления, Transfer-Encoding, HTTPS и современные версии HTTP.

Цель занятия:

Познакомиться с ключевыми механизмами работы HTTP(S), показать эволюцию протокола от HTTP/1.1 к HTTP/2 и HTTP/3 и научиться анализировать сетевые запросы с практической стороны.

Учебные вопросы:

- 1. Перенаправления (3xx).**
- 2. HTTPS и SSL/TLS.**
- 3. Методы HTTP.**
- 4. Эволюция HTTP.**
- 5. HTTP/3 и QUIC.**

1. Перенаправления (3xx).

Перенаправления используются, когда ресурс доступен по другому адресу.

Сервер не отдает сам контент, а сообщает клиенту:

- Код состояния (3xx) — тип перенаправления.
- Заголовок Location — новый адрес ресурса.

После этого клиент (браузер или программа) автоматически делает новый запрос к указанному адресу.

Основные коды и ответы сервера

301 Moved Permanently

- Ресурс навсегда перемещён.
- Сервер отвечает:

```
HTTP/1.1 301 Moved Permanently
Location: https://example.com/new-page
Content-Length: 0
```

- Клиент при следующих запросах может сразу использовать новый адрес.
- Поисковые системы обновляют индекс и «передают вес» ссылок новому URL.

302 Found (временный редирект)

- Ресурс временно доступен по другому адресу.
- Сервер:

HTTP/1.1 302 Found

Location: <https://example.com/temp-page>

- Браузер идёт по новому адресу, но поисковики оставляют исходный URL «главным».

307 Temporary Redirect

- Временный редирект, но с сохранением метода (POST остаётся POST).
- Сервер:

HTTP/1.1 307 Temporary Redirect
Location: /new-api-endpoint

- Используется чаще в API, где важно не потерять метод и тело запроса.

308 Permanent Redirect

- Аналог 301, но также сохраняет метод.
- Сервер:

HTTP/1.1 308 Permanent Redirect

Location: /permanent-endpoint

- Современный стандарт для постоянного переноса

Влияние на кэширование:

- 301, 308 могут кэшироваться браузером и прокси: повторный запрос сразу пойдёт на новый URL.
- 302, 307 считаются временными и обычно не кэшируются.
- Поведение можно уточнять заголовками Cache-Control и Expires.

Влияние на SEO:

- 301, 308 → сигнал «страница переехала навсегда», индекс обновляется, вес ссылок передаётся.
- 302, 307 → поисковики воспринимают как временный перенос, исходный URL остаётся основным.
- Неправильный выбор кода может привести к потере позиций и трафика.

Что такое SEO?

SEO (Search Engine Optimization) — это набор методов и практик, которые помогают сайту занимать более высокие позиции в поисковых системах (Google, Яндекс и др.), чтобы привлечь больше пользователей.

Основные задачи SEO:

1. Техническая оптимизация

- Быстрая загрузка страниц.
- Корректные редиректы (301 вместо 302, если перенос постоянный).
- Настройка HTTPS.

Чистая структура URL.

2. Контентная оптимизация

- Качественные тексты с ключевыми словами.
- Заголовки, метаописания.
- Разметка для сниппетов.

3. Внешняя оптимизация

- Ссылки с других сайтов (ссылочный вес).
- Упоминания бренда.

Зачем фронтендеру знать про SEO?

- Неправильная работа с редиректами может «сломать» SEO (например, 302 вместо 301 → поисковики не передадут ссылочный вес).
- Ошибки в загрузке статики (например, JS или CSS грузятся с лишними редиректами) ухудшают скорость, а скорость — один из факторов ранжирования.
- HTTPS и корректные сертификаты напрямую влияют на доверие поисковиков к сайту.
- SPA-приложения (React, Vue, Angular) часто требуют настройки SSR или специальных заголовков, чтобы поисковики правильно индексировали страницы.

Итог:

- Сервер всегда возвращает код 3xx + заголовок Location.
- Браузер (или клиент) автоматически выполняет новый запрос.
- Тип кода определяет: постоянный ли перенос, кэшировать ли редирект и как реагируют поисковики.

2. HTTPS и SSL/TLS.

Зачем нужен HTTPS?

Обычный HTTP небезопасен:




- данные передаются открытым текстом;
- злоумышленник может перехватить логин, пароль или номер карты;
- можно подменить содержимое страницы (например, через открытый Wi-Fi).

HTTPS решает эту проблему. Это тот же самый HTTP, но он работает поверх защищённого протокола TLS.

Пример:

- HTTP — письмо на бумаге;
- TLS — конверт с замком;
- HTTPS = письмо (HTTP), которое вложили в защищённый конверт (TLS).

Основные цели HTTPS:

- Шифрование 
 - Данные превращаются в набор символов, понятный только вашему браузеру и серверу.
 - Даже если их перехватят, прочитать невозможно.
- Аутентификация 
 - Браузер проверяет, что сайт настоящий, а не подделка.
 - Для этого используются цифровые сертификаты.
- Целостность 
 - Гарантия, что данные не изменились «по дороге».
 - Если кто-то попытается подменить содержимое, браузер заметит ошибку.

Как устанавливается HTTPS-соединение (TLS-рукопожатие):

1. Приветствие клиента (Client Hello)

- Браузер сообщает серверу: «Я поддерживаю такие версии TLS и такие алгоритмы шифрования».

2. Приветствие сервера (Server Hello)

- Сервер выбирает алгоритм.
- Отправляет клиенту цифровой сертификат с публичным ключом.

3. Проверка сертификата

- Браузер проверяет: сертификат выдан доверенным центром (CA), срок действия в порядке, имя сайта совпадает.
- Если что-то не так — браузер показывает предупреждение («Не защищено»).

4. Обмен ключами

- Клиент и сервер договариваются о сессионном ключе (через публичный ключ и криптографию).
- Этот ключ будет использоваться для шифрования всех дальнейших сообщений.

5. Начало защищённой работы

- Дальше идёт обычный HTTP (запросы и ответы), но всё зашифровано.

Сертификаты.

Это «паспорт сайта», который подтверждает его подлинность.

Содержит:

- доменное имя,
- публичный ключ,
- срок действия,
- информацию об издателе (сертификационном центре).

Типы сертификатов:

- DV (Domain Validation) — проверяется только домен.
- OV (Organization Validation) — проверяется компания-владелец.
- EV (Extended Validation) — строгая проверка (обычно используется в банках и финтехе).

Кто выдаёт сертификаты?


Сертификаты выпускают Удостоверяющие центры (CA, Certificate Authority).

Примеры крупных CA:

- DigiCert, Sectigo (бывший Comodo), GlobalSign, GoDaddy.
- Бесплатный CA — Let's Encrypt, которым сегодня пользуются миллионы сайтов.

CA — это «третья сторона», которой доверяют браузеры и ОС.

Как работает доверие (цепочка сертификатов)?

- У каждого браузера и ОС есть список «корневых сертификатов» (root certificates) — это список доверенных СА.
- Когда сайт присылает свой сертификат, браузер проверяет:
 - Подписан ли он корневым СА или промежуточным центром, который в итоге восходит к корневому?
 - Если да — всё ок .
 - Если нет (самоподписанный сертификат, неизвестный СА) — браузер покажет предупреждение.

Пример цепочки:

[Root CA] → [Intermediate CA] → [Сертификат сайта]

Самоподписанные сертификаты.

- Сервер может выпустить сертификат сам, без СА.
- Такой сертификат работает технически, но браузеры ему не доверяют, так как подписи «третьей стороны» нет.
- Используется обычно для тестирования или внутри компании.

Ключевые выводы:

- HTTPS = HTTP, но через зашифрованный канал.
- TLS нужен, чтобы защитить:
- данные (шифрование),
- личность сервера (аутентификация),
- содержимое от подмены (целостность).
- Сертификаты создают доверие: браузер верит только тем, кто получил «подпись» от надёжных центров.

👉 В DevTools → Security можно посмотреть, каким сертификатом защищён сайт и какие алгоритмы используются.



2 HTTP 1.0



DevTools is now available in Russian

[Don't show again](#)

Elements Console Sources Network Performance

Privacy



Controls



Third-party cookies

Security



Overview

Main origin

Secure origins

<https://api.rivox.io><https://api.sales-ninj...><https://cdn.carrotque...><https://api.carrotque...><https://ru.hexlet.io>

Security overview



This page is secure (valid HT



Certificate - valid and trusted

The connection to this site is us
by WE1.[View certificate](#)

Connection - secure connection

The connection to this site is en
X25519MLKEM768, and AES_12

Инструмент просмотра сертификатов: hexlet.io

Общие

Подробнее

Выдан:

Общее имя (ЦС)

hexlet.io

Организация

<Не является частью сертификата>

Подразделение

<Не является частью сертификата>

Выдан:

Общее имя (ЦС)

WE1

Организация

Google Trust Services

Подразделение

<Не является частью сертификата>

Срок действия

Дата выдачи

четверг, 31 июля 2025 г. в 17:18:28

Срок действия истекает

среда, 29 октября 2025 г. в 18:18:12

Цифровые отпечатки сертификата
с подписью SHA-256

Сертификат

753ef1cd3a0ae13c9d1f86f7291a892352517ff25
4917f354a82e86472edda7b

Открытый ключ

82c440f0315b4baa9cefaaab1125aaf7f1e94df4f
9fa01acb8d93ce101893969

3. Эволюция HTTP: новые возможности HTTP/2

HTTP/1.1 был стандартом с конца 90-х, но у него есть серьёзные проблемы:

- Один запрос = одно соединение:
 - В старых браузерах было ограничение: максимум 6 соединений к одному домену.
 - Если на странице много ресурсов (картинок, скриптов, CSS), они загружались медленно.
- Head-of-line blocking (блокировка по голове очереди):
 - Если один запрос завис, все следующие в том же соединении тоже ждали.
- Большие заголовки:
 - Каждый запрос передаёт повторяющиеся заголовки (cookies, user-agent и т. д.).
 - Это лишний трафик и задержки.
- Костыли для ускорения:
 - Разработчики придумывали хаки: объединение файлов (CSS/JS), спрайты для картинок, шардирование доменов (разносить ресурсы на cdn1, cdn2 и т. п.).

HTTP/2 появился в 2015 и решил многие проблемы.

◆ Мультиплексирование

- Несколько запросов могут идти по одному соединению одновременно.
- Нет блокировки — браузер может загружать десятки ресурсов параллельно.

◆ Бинарный протокол

- В HTTP/1.1 заголовки и тело — текстовые (читаемые человеком).
- В HTTP/2 всё кодируется в бинарных фреймах, что быстрее и эффективнее для машин.

◆ Сжатие заголовков (HPACK)

- Повторяющиеся заголовки не отправляются каждый раз.
- Экономия трафика, особенно для API с большим количеством однотипных запросов.

◆ Сервер Push (опционально)

- Сервер может сам отправить ресурсы (например, CSS или JS), ещё до того, как браузер их запросит.
- Используется редко, так как иногда перегружает сеть.

Выводы:

- HTTP/1.1 = медленный, «текстовый» протокол с ограничениями.
- HTTP/2 = более быстрый, бинарный, эффективный (особенно при множестве мелких запросов).
- Для фронтенд-разработчика это значит:
- нет нужды в спрайтах и агрессивной конкатенации файлов;
- ресурсы можно хранить отдельно (это облегчает разработку и кэширование).

👉 В DevTools (Network → Protocol) можно посмотреть, какой протокол использует сайт: http/1.1 или h2.

Знакомимся с целями и задачами курса

2 HTTP 1.0

теория

тесты



DevTools is now available in Russian

[Don't show again](#)[Always match Yandex Browser's language](#)[Switch DevTools to Russian](#)

Elements

Console

Sources

Network

Performance

Memory

Application

Privacy and security

Lighthouse

Recorder

☐ Preserve log☐ Disable cache

No throttling



Filter

☐ Invert

More filters ▼

100,000 ms

200,000 ms

300,000 ms

400,000 ms

500,000 ms

600,000 ms

700,000 ms

800,000 ms

900,000 ms

1,000,000 ms

1,100,000 ms

1,200,000 ms

1,300,000 ms

Name	Status	Protocol	Type	Initiator
modeled-conversion	200	h2	fetch	userBundle.js:1
25559621?wv-part=5&wv-check=9...	200	h2	fetch	tag.js:99
prolong	200	h2	fetch	userBundle.js:1
manifest.webmanifest	200	h3	manifest	Other
manifest.webmanifest	200	h3	manifest	Other
manifest.webmanifest	200	h3	manifest	Other
manifest.webmanifest	200	h3	manifest	Other

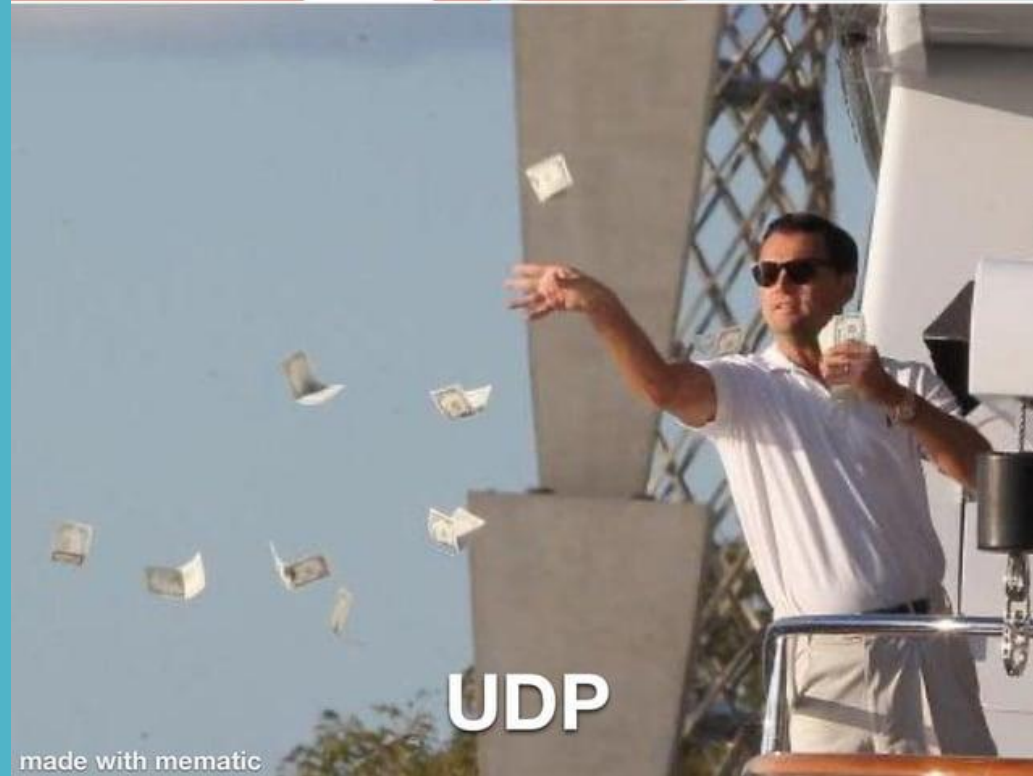
4. HTTP/3 и QUIC.

Немного о TCP и UDP.

TCP — протокол с установлением соединения и гарантией доставки и в правильном порядке.

«Аккуратный почтальон»: проверяет, чтобы все письма пришли, и в правильном порядке. Но если одно письмо потерялось — задерживаются все остальные.

UDP — протокол с без установления соединения и без гарантии доставки или порядка. «Быстрый гонец»: шлёт пакеты как есть. Если что-то потерялось — едет дальше. Быстрее, но без гарантии порядка и доставки.



Проблема HTTP/2:

- Работает поверх TCP.
- Если один пакет потерялся → все запросы ждут его (задержка).
- Особенно плохо в мобильных сетях и при плохом Wi-Fi.




Решение: QUIC и HTTP/3:

- QUIC — новый транспорт от Google, основанный на UDP.
- Внутри сразу встроено шифрование (TLS 1.3).
- HTTP/3 работает поверх QUIC.

Что делает QUIC?

- QUIC берёт скорость UDP, но добавляет «умный слой над ним»:
- Гарантированная доставка — если пакет потерялся, QUIC договаривается и запрашивает только этот пакет (а не блокирует всё соединение, как TCP).
- Порядок пакетов — внутри QUIC потоки независимы, и каждый сам восстанавливает свой порядок.
- Шифрование встроено сразу — используется TLS 1.3 по умолчанию.

Преимущества HTTP/3:

-  Нет стопора из-за одного пакета — остальные запросы продолжают выполняться.
-  Быстрое подключение — меньше «рукопожатий», сайт открывается быстрее.
-  Стабильность в мобильных сетях — соединение не обрывается при переключении с Wi-Fi на LTE.

Поддержка:

- Работает во всех современных браузерах (Chrome, Firefox, Safari, Edge).
- Поддерживается на сайтах Google, YouTube, Facebook, Cloudflare и др.
- В DevTools (Network → Protocol) видно h3, если сайт использует HTTP/3.

Итог:

- HTTP/1.1 — старый, медленный.
- HTTP/2 — быстрый, но зависит от TCP (страдает при потерях).
- HTTP/3 — ещё быстрее и надёжнее, особенно в нестабильных сетях.

Итоги лекции:

- Перенаправления (3xx): сервер шлёт код + Location, бывают постоянные (301/308) и временные (302/307).
- HTTPS / TLS: шифрование, аутентификация, целостность; сертификаты от доверенных СА.
- HTTP/1.1 → HTTP/2 → HTTP/3:
 - 1.1 — медленный, блокировки.
 - 2 — мультиплексирование, бинарный формат, сжатие заголовков.
 - 3 — QUIC, быстрый и устойчивый к потерям.

Контрольные вопросы:

- Чем отличаются коды 301 и 302?
- Какие задачи решает HTTPS?
- Что такое TLS-рукопожатие в упрощённом виде?
- Какие проблемы HTTP/1.1 решает HTTP/2?
- Почему HTTP/3 работает поверх UDP и какие преимущества это даёт?

Домашнее задание:

1. https://ru.hexlet.io/courses/http_protocol

Материалы лекций:

<https://github.com/ShViktor72/Education2025>

Обратная связь:

colledge20education23@gmail.com