NodeJS

```
let lesson = {
  id:       '11'
  themes:   ['Auth',
'JWT', 'Security'],
  date:     '17.12.2024',
  teacher: {
    name:   'Николай Лапшин'
  }
};
```

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Slack
#javascript-2022-09
или #general

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Тема вебинара

Аутентификация и безопасность

Николай Лапшин
Powertech. Tech Lead/Principal Engineer

О себе:
- в IT 10+ лет.
- Последнии несколько лет Tech Lead в Digital Advertising High-Load проекте
- Пишу на TS/Golang. Использую PostgreSQL и MongoDB.
- В OTUS - 3 года, более 150 вебинаров и более 1000 студентов.

- преподаватель курса Node.JS/MongoDB/PostgresDBA/NoSQL и другие в OTУС

Блог: https://medium.com/@nlapshin1989

Маршрут вебинара

- Аутентификация
- Токены
- Полезные пакеты
- Рефлексия

Цели вебинара

К концу занятия вы сможете

1. Разобраться с теорией аутентификации
2. Узнать, как настроить аутентификацию в Node.JS
3.

Скажите пару слов

обсуждение знаний

1. Какие задачи безопасности возникают в Web-приложении?
2. Какие способы аутентификации знаете?
3.

Аутентификация

Авторизация и аутентификация

- Аутентификация - процесс проверки, действительно пользователь тот, за кого он себя выдает. Как правило, это 401 HTTP статус в случае ошибки.
- Авторизация - процесс определения, какие действия или ресурсы разрешены для пользователя, который уже прошел аутентификацию. Как правило, это 403 HTTP статус в случае ошибки.

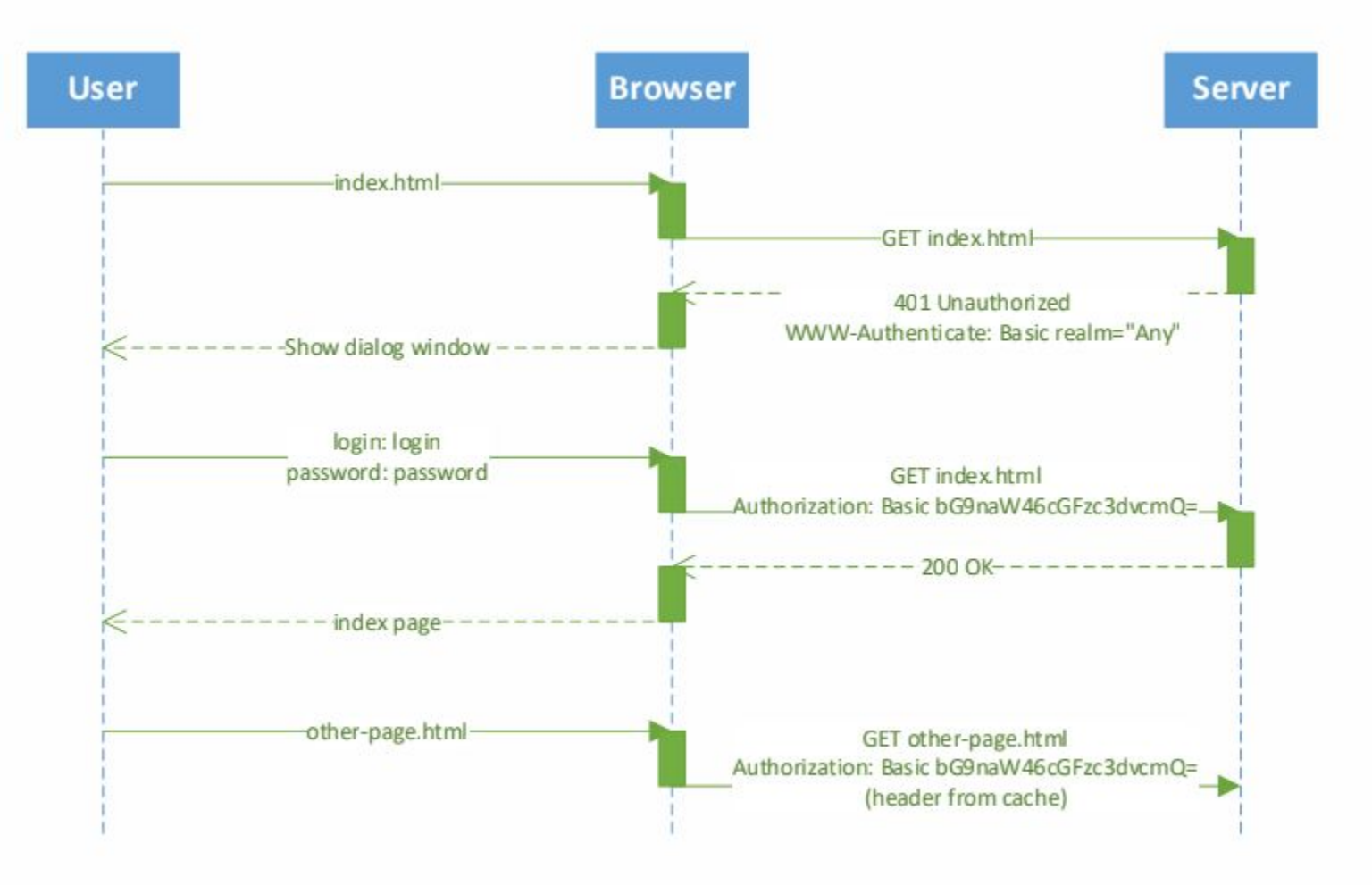
Аутентификация

- Аутентификация — это процесс проверки подлинности личности пользователя. Цель - определить, что пользователь тот, за кого он себя выдает.
- Способы аутентификации: через пароль, биометрия, токены доступа, многофакторная аутентификация(MFA).

Аутентификация. Виды.

- Basic Auth.
- Form-Based Auth.
- OAuth.

Аутентификация. Basic Auth.



Аутентификация. Basic Auth. Плюсы и минусы.

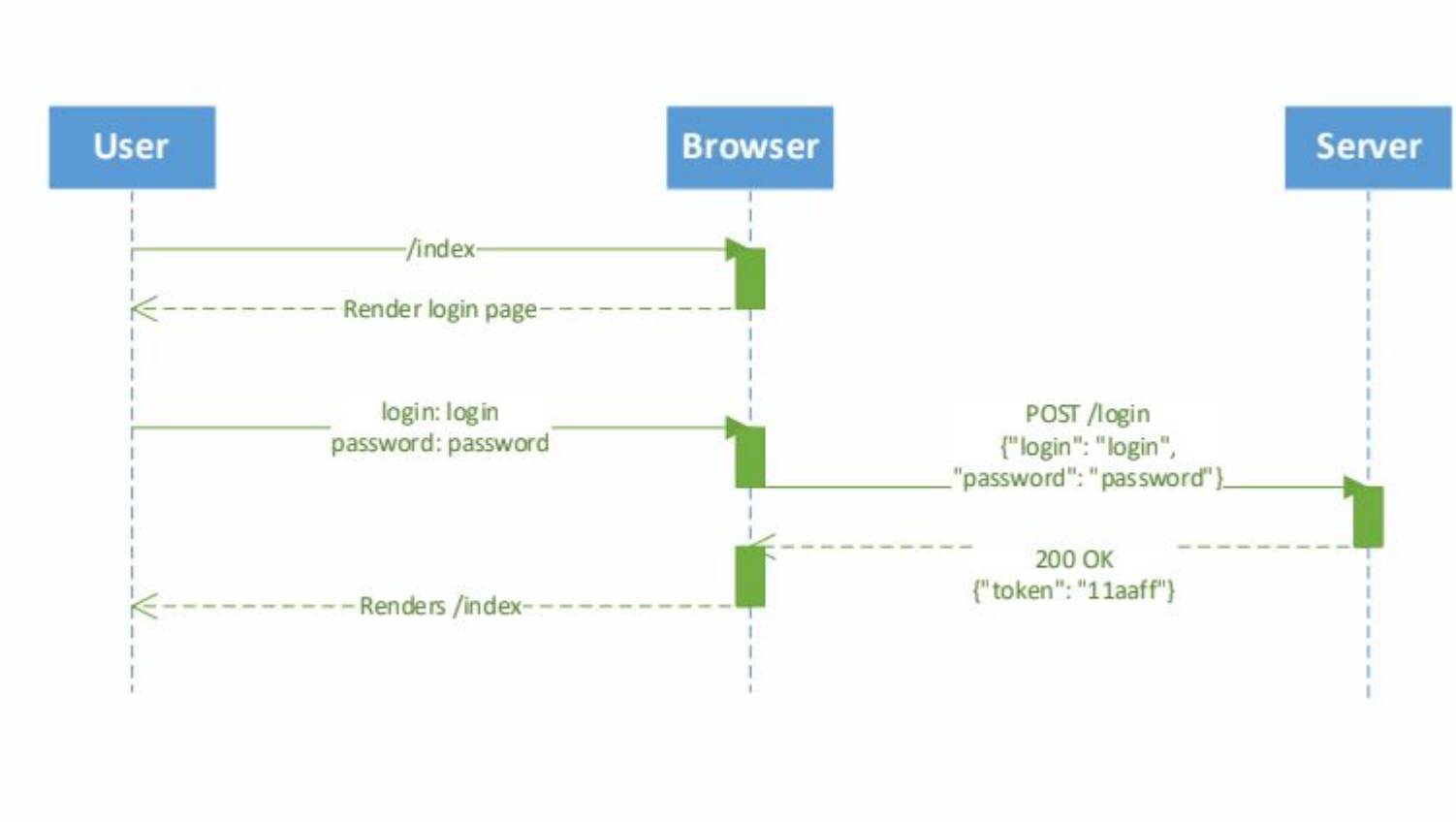
Плюсы

- Встроенная поддержка браузера.
- Какой-никакой стандарт
- Идеальная для веб-сервисов
- Никаких cookies
- Простая схема, передается в заголовке

Минусы

- Без HTTPS – мы просто отдаём логин-пароль злоумышленнику
- Невозможная кастомизация логина (SMS, прочее)
- Logout?

Аутентификация. Form-Based Auth.



Аутентификация. Form-Based Auth. Как работает?

- Пользователь через веб-форму вводит логин и пароль.
- Отправка данные на специальный HTTP endpoint.
- Endpoint проверяет верность логина и пароля.
- Выдается токен доступа(обычно хэш с ограниченным временем жизни)
- Пользователь использует его для входа пока он не “протухнет”.

Аутентификация. Form-Based Auth. Плюсы и минусы.

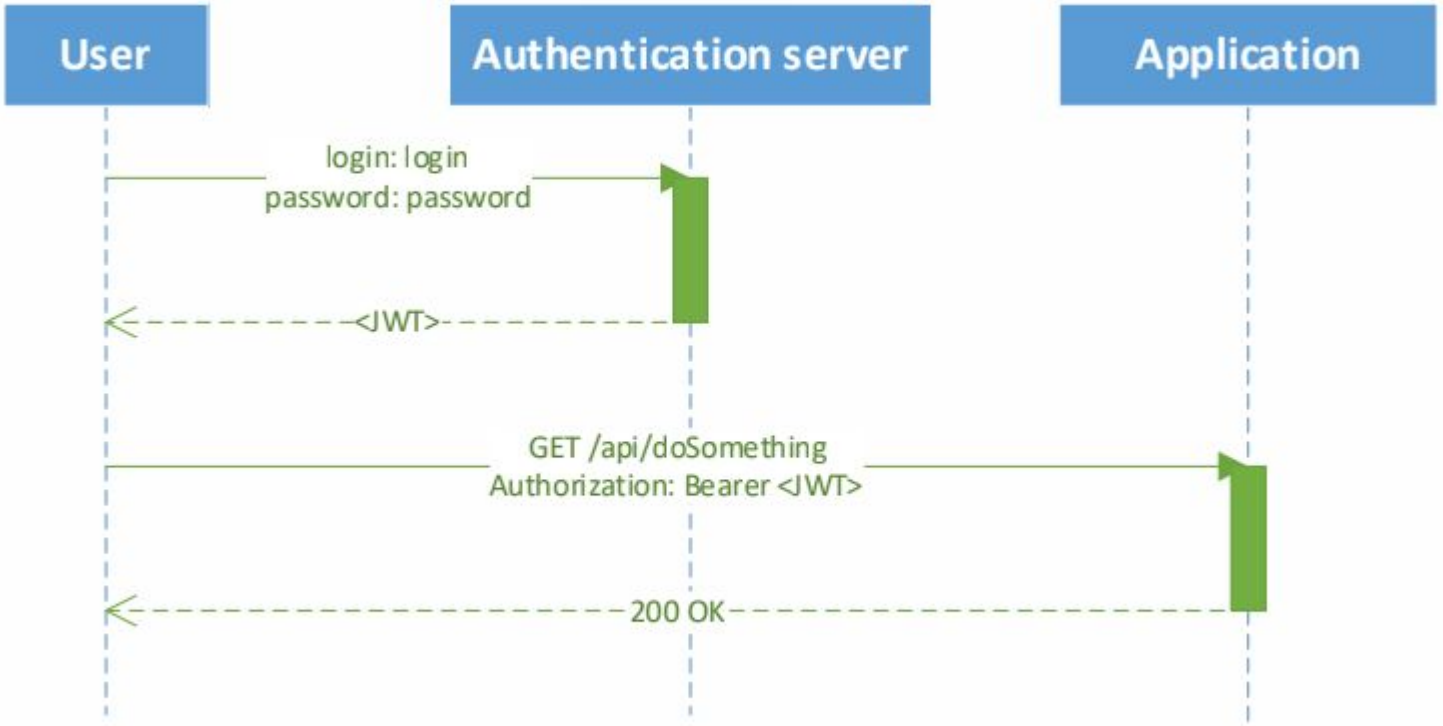
Плюсы

- Кастомизация для веб-приложений
- Самая популярная схема
- Логаут лёгкий – удаляем куку

Минусы

- Обычно для веб-сервисов как-то отдельно пропускают логин и сразу выдают токен
- Без HTTPS – мы просто отдаём логин-пароль злоумышленнику
- Необходима ещё и «вторичная» аутентификация
- Логаут на всех устройствах – не так просто сделать

Аутентификация. OAuth.



Аутентификация. OAuth. Как работает?

- **OAuth (Open Authorization)** — это стандартный протокол для авторизации, который позволяет сторонним приложениям получать ограниченный доступ к ресурсам пользователя без необходимости передачи паролей
- Регистрируемся в одном из популярных сервисов(или провайдеров авторизации), например, Yandex.
- Когда пользователь пытается авторизоваться, его перенаправляют на страницу авторизации провайдера.
- Если пользователь соглашается, то приложение выдает код авторизации и возвращает его назад.
- Финально выдается токены доступа - access и refresh.

Аутентификация. OAuth. Плюсы и минусы.

Плюсы

- Не нужно разворачивать свою систему хранения пользовательских данных, делегируем другим сервисам.
- Удобно для пользователя, не нужно заводить под каждый сервис пароли.
- Стандартизация и современный протокол.

Минусы

- Сложность реализации.
- Зависимость от провайдера.

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Токены и сессии

Токены и сессии. Виды

- Случайно сгенерированные токены (Persistent Tokens)
- Hash Based Token.
- JWT(продвинутый Hash Based)

Случайные токены

- Генерируем случайный токен(без payload).
- Храним в базе данных.
- При запросе сравниваем с токен в базе данных

Пример: crypto.randomBytes(32).toString('hex');

Hash Based токены

- Генерируем случайный токен с нагрузкой пропущенные через hash функцию.
- Аутентификация через сравнение хэшей.

Пример: crypto.createHash('sha256').update(token).digest('hex');

JWT токены

JWT (JSON Web Tokens) — простой и безопасный способ передачи информации между клиентом и сервером.

JWT состоит из трех частей, разделенных точками ('.'):

1. **Header (Заголовок)**
2. **Payload (Полезная нагрузка)**
3. **Signature (Подпись)**

Пример: jwt.sign(payload, SECRET_KEY, { expiresIn: '1h' });

JWT токены. Из чего состоит.

Заголовок обычно состоит из двух частей: типа токена (JWT) и алгоритма подписи (например, HMAC SHA256 или RSA).

Пример: { "alg": "HS256", "typ": "JWT" }

Полезная нагрузка содержит данные, которые можно передавать на клиент - это или зарезервированные слова(iss, exp и прочие) и данные, пользователя(username, id и т.д.)

Подпись создается с использованием заголовка, полезной нагрузки и секретного ключа (или приватного ключа). Подпись необходима для проверки подлинности токена и целостности данных.

https://jwt.io/

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Хранение паролей

Хранение паролей

- Хранить пароли в открытом виде небезопасно.
- Необходимо хранить пароли в виде хэша
- Для хэширования необходимо использовать современные библиотеки и алгоритмы

Хранение паролей. Пакеты.

- Встроенные пакет crypto.
- Популярный пакет bcrypt.
- Один из лучших алгоритмов argon2

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Библиотеки Node.JS

Helmet

- Существует множество популярных web уязвимостей(XSS и т.д.)
- Библиотека helmet позволяет защитить от этого

Passport

- **Passport.js** — это гибкая и модульная библиотека для аутентификации в Node.js. Она поддерживает множество стратегий аутентификации, таких как локальная аутентификация
- Стратегии аутентификации JWT
- Интеграция с Oauth (facebook, Google и т.д.)
- Отлично работает с express, NestJS и т.д.

Рефлексия

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то, что узнали на вебинаре?

Остались вопросы от материала?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Тема вебинара

Приходите на следующий вебинар