**Протоколы Http и Https** – протоколы прикладного уровня для передачи данных (изначально применялся только к HTML документам) от клиента к серверу и обратно.

**Структура http сообщения** – состоит из стартовой строки, заголовков, тела.

Стартовая строка – строка, содержащая основную информацию о запросе и ответе.

Заголовок – Его содержание характеризует тело сообщения и параметры его передачи и др. метаинформацию.

Заголовки и тело разделены пустой строкой.

Тело – содержит данные, передаваемые между клиентом и сервером.

Обязательными структурами http сообщения являются стартовая строка и заголовок со значением host (т.к. преобразование доменного имени в ip адрес происходит на стороне клиента и сервер не обладает информацией к какому доменному имени обратились).

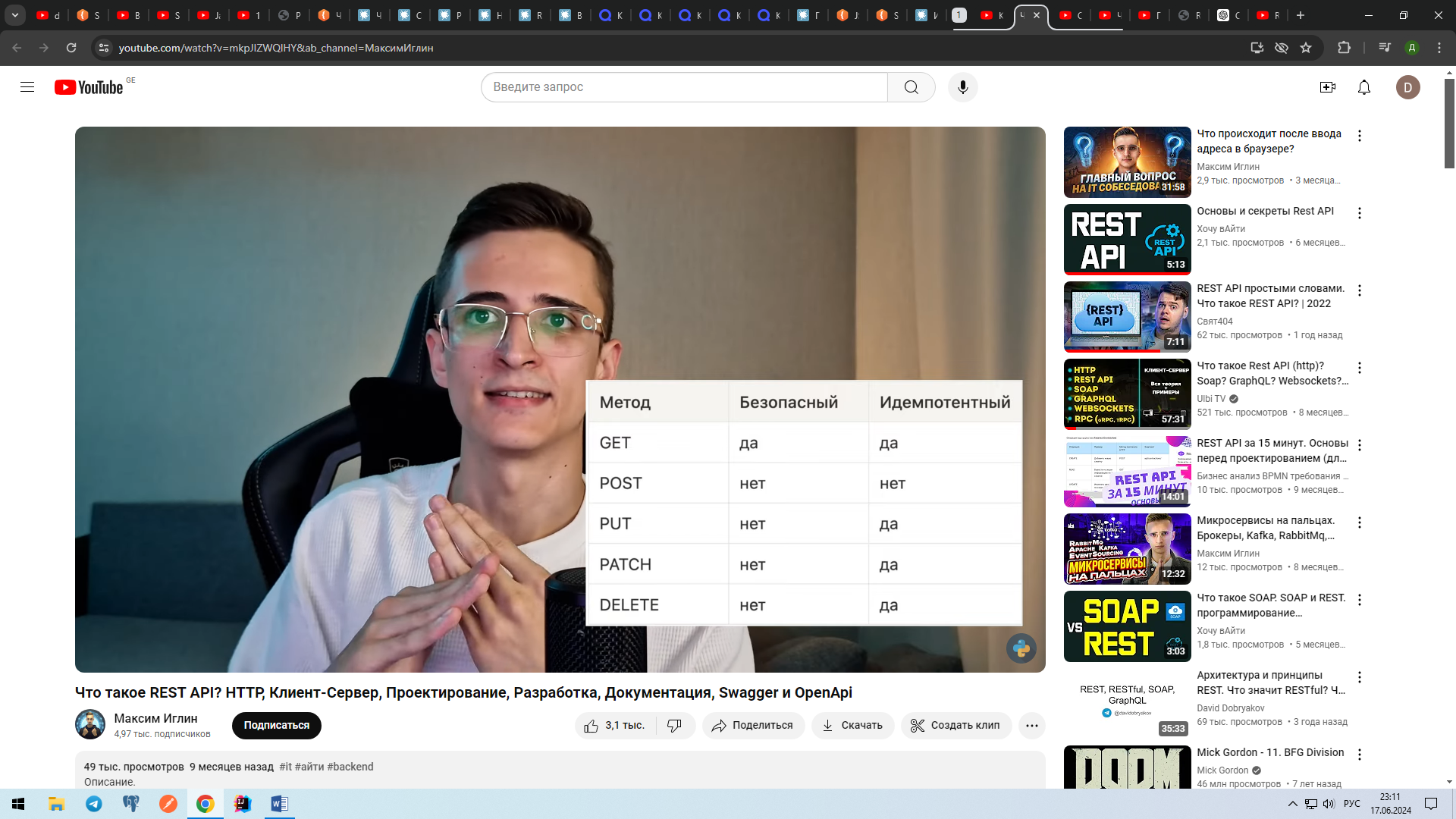
|  |  |  |
| --- | --- | --- |
| Пример | Запрос по http://example.com/me | Ответ |
| Стартовая строка | GET/me HTTP/1.0 | HTTP 1.0 200 OK |
| Заголовок | Host: example.com | \*Заголовки\* |
| Тело | \*Тело запроса\* | \*Пересылаемый ответ\* |

|  |  |  |
| --- | --- | --- |
|  | *Запрос* | *Ответ* |
| Стартовая строка | Содержит метод, URI, версия HTTP | Содержит версию HTTP, код состояния, сообщение |
| Заголовки | Общие заголовки - Применяется как для запроса, так и для ответа. | Cache-Control: Управляет кэшированием как клиентом, так и прокси-серверами.  Connection: Управляет аспектами управления соединением, например, поддержкой соединения открытым.  Date: Указывает дату и время, когда сообщение было отправлено.  Pragma: Используется для передачи директив, связанных с управлением кэшированием.  Trailer: Указывает, какие заголовки следуют после тела сообщения в формате chunked.  Transfer-Encoding: Указывает, какие типы преобразований были применены к телу сообщения, чтобы сделать его безопасным для передачи.  Upgrade: Указывает, что клиент хочет переключиться на другую протокольную версию.  Via: Указывает на промежуточные прокси-серверы, через которые прошел запрос или ответ.  Warning: Передает дополнительную информацию о статусе или преобразованиях сообщения. |
| Заголовок запроса клиента - Применяется только для запроса. | Accept: Указывает, какие типы медиа (MIME-типы) клиент может обрабатывать.  Accept-Charset: Указывает наборы символов, которые клиент может обработать.  Accept-Encoding: Указывает типы контент-кодирования, которые клиент может понять.  Accept-Language: Указывает языки, которые клиент предпочитает.  Authorization: Передает учетные данные для аутентификации пользователя на сервере.  Host: Указывает доменное имя сервера и (необязательно) номер порта, к которому клиент хочет подключиться.  If-Range: Выполняет запрос диапазона, если ресурс не изменился; иначе возвращает весь ресурс.  If-Modified-Since: Выполняет запрос только в случае, если ресурс изменился с указанного времени.  Range: Запрашивает определенный диапазон байтов ресурса.  Referer: Указывает URL, откуда был сделан запрос.  User-Agent: Передает информацию о клиенте, осуществляющем запрос (например, браузере или приложении)  И др. |
| Заголовок ответа сервера - Применяется только для ответа. | Accept-Ranges: Указывает, поддерживает ли сервер запросы диапазонов байтов.  Age: Указывает возраст объекта в кэше прокси-сервера.  ETag: Передает уникальный идентификатор версии ресурса (проверка актуальности кэшируемого ресурса)  Location: Указывает URL для перенаправления клиента (например, при статусе 3xx).  Proxy-Authenticate: Запрашивает у клиента учетные данные для аутентификации на прокси-сервере.  Retry-After: Указывает время, через которое клиент может повторить запрос.  Server: Передает информацию о серверном программном обеспечении.  Set-Cookie: Отправляет куки для сохранения клиентом.  Vary: Указывает, какие заголовки запроса использовались для выбора контента.  WWW-Authenticate: Запрашивает у клиента учетные данные для аутентификации на сервере. |
| Заголовок сущности - Определяет метаданные сущности. Если сущности нет, то метаданные URI запроса. | Allow: Указывает методы HTTP, поддерживаемые ресурсом.  Content-Encoding: Указывает кодировку, примененную к телу сообщения.  Content-Language: Указывает язык, используемый в теле сообщения.  Content-Length: Указывает длину тела сообщения в байтах.  Content-Location: Указывает альтернативный URL для тела сообщения.  Content-MD5: Указывает MD5 хэш тела сообщения для проверки целостности.  Content-Range: Указывает, какая часть полного ресурса передается.  Content-Type: Указывает MIME-тип тела сообщения.  Expires: Указывает дату и время, после которых ресурс считается устаревшим и его нельзя использовать из кэша.  Last-Modified: Указывает дату и время последнего изменения ресурса. |
| Тело | Содержит передаваемый контент | |

*URI* – путь до конкретного ресурса на сервере.

*Методы* – определяют действие, которое клиент хочет выполнить на стороне сервера.

|  |  |
| --- | --- |
| GET | Получение ресурса с сервера  Помимо обычного GET-запроса имеется также условные Get-запросы и частичные GET-запросы.  Условный - получение ресурса только в том случае, если ресурс изменился с последнего запроса клиента (с помощью заголовков If-Modified-Since или "f-None-Match) или использовать кэшированную версию.  Частичный – получение части ресурса с помощью определения диапазона байтов. |
| POST | Создание нового ресурса на сервере |
| HEAD | как GET, только сервер не отправляет тело, только заголовки (проверка существования ресурса или актуализация URL адреса) |
| PUT | обновление ресурса |
| PATCH | обновление фрагмента ресурса |
| DELETE | удаление ресурса |
| TRACE | позволяет увидеть путь прохождения запроса + его изменения |
| CONNECT | подключение через proxy |
| OPTIONS | определение возможностей сервера (какие методы и протоколы он поддерживает) |

*Безопасные и идемпотентные методы:*

Метод удовлетворяет критерию безопасности, если он не изменяет состояние ресурсов на сервере.

Метод удовлетворяет критерию идемпотентности, если его многократное использование приводит к такому же состоянию ресурса, если бы метод вызывался один раз, при этом клиент может получать разные ответы.

**Коды состояния** – код, указывающий на результат обработки запроса. Информационное (100), успешное выполнение (200), перенаправление (300), ошибка на стороне клиента (400), ошибка на стороне сервера (500)

Сообщение – краткое текстовое описание кода состояния.

**Отличия протоколов**

|  |  |  |
| --- | --- | --- |
|  | HTTP | HTTPS |
| Безопасность | Не предполагает шифрование информации | Передает данные в нижестоящий протокол криптографический протокол ssl (устарел) или tls для обеспечения конфиденциальности передаваемых данных. Обеспечивает:  Шифрование данных (защита данных с помощью закрытого ключа) – предотвращает чтение перехваченных данных.  Сохранность данных (проверка целостности данных с помощью хэш-функции) – предотвращает возможность манипулирования данными во время передачи.  Аутентификация (защита от перенаправления) – см. сертификация. |
| Порт | 80 | 443 |
| Сертификация | Не имеет | Сервер аутентифицируется с помощью сертификатов, выданных специальными удостоверяющими центрами. Сертификация позволяет избежать атаки типа man in the middle.  Трехстороннее рукопожатие:  Установка соединения (обмен приветственными сообщениями с параметрами TLS соединения).  Аутентификация и обмен ключами - Сервер передает свой сертификат клиенту, содержит публичный ключ и информацию о удостоверяющем центре. Клиент обращается к центру сертификации, чтобы убедиться в подлинности.  Независимое вычисление закрытого ключа. |

**MIME type –** спецификация для передачи по сети файлов различного типа, которая указывает на тип и формат передаваемых данных в теле http. Указывается в заголовке Content-Type, состоит из type/subtype;parameter=value. Основные типы: application (бинарные данные любого вида), audio, example (для примеров), font (шрифты), image, model (описание 3d моделей), text, video.

**Версии http**

**Http 0.9 –** функционал был ограничен наличием метода GET, который возвращал только HTML страницу(отсутствовали заголовки, невозможно передавать данные отличные от HTML).

**Http 1.0** – Добавилась поддержка методов POST и HEAD, расширены типы передаваемого контента MIME-types, добавлены заголовки и коды состояния, базовая поддержка кэширования, при этом возможны только одноразовые соединения (клиент отправляет запрос, получает ответ и соединение закрывается).

**Http 1.1** – Добавлены новые методы, поддержка cookie, расширены возможности кэширования, поддержка постоянного соединения (дает возможность отправки нескольких запросов без пересоздания соединения), пайплайнинг запросов (возможность отправки нескольких запросов, не дожидаясь ответа на предыдущие), для его использования нужно точно знать границы сообщений (используются заголовки content-length или Transfer-encoding: chunked)

Cookie – служебная информация, которую сервер отправляет клиенту для ее сохранения и последующей передачи для идентификации этого клиента и его предыдущей активности, т.к. сервер не запоминает состояний (при авторизации пользователя на сайте сервер отправляет id сессии и при последующих запросах клиенту не требуется повторная авторизация)

По-прежнему актуальна проблема блокировки очереди запросов (если предыдущий медленный запрос блокировал последующие)

**Http 2.0** – Не изменилась общая семантика протокола, но изменился способ упаковки и передачи между клиентом и сервером, но сам набор данных остается прежним (для совместимости с протоколом 1.1). Рекомендуется использовать https соединение (не является стандартом).

*Бинарный формат вместо текстового –* представление http запроса в бинарном формате, а не в текстовом. *П*озволяет уменьшить объем передаваемой информации приходящейся на использование текстового формата.

*Физическое разделение на фреймы -* Использование физического разделения http запроса на фреймы (больше не используется логическое разделение на стартовую строку, заголовки и данные, которые по сути являются одним текстовым сообщением). Фрейм состоит из технического заголовка (тип фрейма, размер, флаги, идентификатор стрима) и данных, которые он должен передать.

*Мультиплексирование* – создание множества двунаправленных каналов между сервером и клиентом, в рамках одного соединения.

Позволяет отправлять несколько запросов одновременно (не последовательно, как в 1.1) и распараллеливать задачу отправки запроса и ответа.

*Server push –* принудительное наполнение кэша клиента данными не дожидаясь инициирующего воздействия со стороны клиента (отправка фрейма PUSH\_PROMISE) – можно реализовать логику работы сервера таким образом, что клиенту не придется создавать запрос на отправку ему какого-либо ресурса, т.к. сервер отправит его превентивно. Например, при отправке запроса на html страницу сервер может сразу же отправить ресурсы, которые включены в этот документ (js ресурсы, стили, изображения), что не требует формирование запроса клиентом.

*Сжатие заголовков методом HPACK –* Используется индексация заголовков, т.е. передается не символьное обозначение заголовка, а индекс соответствующий этому заголовку. Значение этих заголовков хранится в статической (часто повторяющиеся заголовки) и динамической таблицах, как на стороне сервера, так и на клиентской стороне для сопоставления индексов их значениям. Для значений этих заголовков используется сжатие методом Хаффмана.

Также уменьшает объем передаваемой служебной информации.

*Приоритизация запросов и выстраивание дерева зависимостей –* позволяет гибко управлять ресурсами сервера на основе полученной информации.