**Протокол передачи гипертекста (Hypertext Transfer Protocol - HTTP) -** это прикладной протокол для передачи гипертекстовых документов, таких как HTML. В настоящий момент используется для передачи произвольных данных. Основан на TCP/IP (TLS – защищенный TCP), использует его для пересылки своих сообщений. Может использоваться для передачи содержимого серврам (например с помощью HTML-форм), а также для получения частей документа, с целью обновления веб-страницы по запросу (AJAX).

Гипертекст - термин, обозначающий систему из текстовых страниц, имеющих перекрёстные ссылки. (Текст с навигацией)

Протокол следует **клиент-серверной** модели. Основынми реализациями клиентов являются браузеры. Основными реализациями серверов – Apache, nginx, Google Web Server.

Клиенты и серверы взаимодействуют обмениваясь одиночными сообщениями, а не потоком данных. Сообщения, отправленные клиентом, называются **запросами**, а сообщения, отправленные сервером – **ответами**.

Для взаимодействия с сервером клиент выполняет следующие шаги:

1. Открытие TCP соединения, которое будет использоваться для отправки запроса и получения ответа. Клиент может открыть новое соединение, переиспользовать существующее или открыть несколько TCP-соединений.
2. Отправка запроса в виде HTTP-сообщения.
3. Чтение ответа от сервера.
4. Закрывает или переиспользует соединение дальнейших запросов.

Сервер не обязательно расположен на одной машине, и наоборот - несколько серверов могут быть расположены (поститься) на одной и той же машине. В соответствии с версией HTTP/1.1 и имея [Host](https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Host) заголовок, они даже могут делить тот же самый IP-адрес.

HTTP – это протокол **без сохранения состояния.**Не существует связи между двумя запросами, которые последовательно выполняются по одному соединению. Сервер не сохраняет никаких данных о промежуточном состоянии между парами “запрос ответ”.

HTTP **расширяемый.**  Заголовки (headers) сделали этот протокол легким для расширения. Новая функциональность может быть даже введена простым соглашением между клиентом и сервером о семантике нового заголовка.

**HTTP-сообщения**

Каждое HTTP-сообщение состоит из следующих частей:

1. Стартовая строка, определяющая тип сообщения
2. HTTP заголовки
3. Пустая строка, отделяющая заголовки от тела
4. Тело сообщения – данные запроса (ответа)

Заголовки и тело могут отсутствовать, стартовая строка является обязательным элементом.

**Стартовая строка запроса** состоит из следующих элементов

1. Метод HTTP, указывающий какое действие выполнить для данного ресурса.
2. URI – унифицированный идентификатор ресурса
3. Версия HTTP.

**Стартовая строка ответа (строка статуса)** содержит следующую информацию:

1. Версию протокола, обычно HTTP/1.1
2. Код состояния (status code), сообщающий об успешности запроса или причине неудачи.
3. Пояснение (status text). Краткое текстовое описание кода состояния.

HTTP/1.1 404 Not Found.

Заголовки HTTP – пара имя – значение. Позволяют клиенту и серверу отправлять дополнительную информацию с HTTP запросом или ответом.

Они могут содержать описание данных и информацию, необходимую для взаимодействия между клиентом и сервером. Имя заголовка не чувствительно к регистру, пробелы перед значением игнорируются.

**Виды заголовков**

***Основные заголовки (General headers*)** – должны включаться в любое сообщение клиента и сервера.

**Cache-Control** используется для задания инструкций кеширования как для запросов, так и для ответов.

*Инструкции кеширования для запросов*

* **Cache-Control: max-age=<seconds>** - минимальное время жизни (относительно времени запроса) в течении которого ресурс будет считаться актуальным;
* **Cache-Control: max-stale[=<seconds>]** - указывает, что клиент хочет получить ответ, для которого было превышено время устаревания. seconds указывает что ответ не должен быть просрочен более чем на указанное значение;
* **Cache-Control: min-fresh=<seconds>** - указывает, что клиент хочет получить ответ, который будет актуален как минимум указанное количество секунд;
* **Cache-Control: no-cache**  - указывает на необходимость отправить запрос на сервер, для проверки актуальности ресурса;
* **Cache-Control: no-store** - Кеш не должен хранить информацию о запросе и ответе;
* **Cache-Control: no-transform** - Прокси могут в каких-либо целях преобразовывать заголовки и даже тело отклика. no-transform запрещает это;
* **Cache-Control: only-if-cached** - указывает на необходимость использования только закешированных данных. Запрос на сервер не должен посылаться.

*Инструкции кеширования для ответов*

* **Cache-Control: must-revalidate -** просроченные ресурсы из кеша не должны быть использованы (даже при отсутсвии интернета)
* **Cache-Control: no-cache**
* **Cache-Control: no-store**
* **Cache-Control: no-transform**
* **Cache-Control: public –** указывает что ответ может быть закеширован в любом кеше (как в браузере так и в прокси).
* **Cache-Control: private –** разрешает кеширование только в однопользовательских кешах (т.е. в браузерах).
* **Cache-Control: proxy-revalidate -** То же самое, что must-revalidate, но применимо только к разделяемым кешам (например, прокси) и игнорируется частными кешами.
* **Cache-Control: max-age=<seconds>**
* **Cache-Control: s-maxage=<seconds> -** Переопределяет max-age или заголовок Expires, но применяется только для разделяемых кешей (например, прокси) и игнорируется частными кешами.

Заголовок **Connection** определяет, остаётся ли сетевое соединение активным после завершения текущей транзакции (запроса).

keep-alive – не закрывать TCP соединение.

**Close –** указывает, что клиент или сервер не хотели бы закрывать соединение.

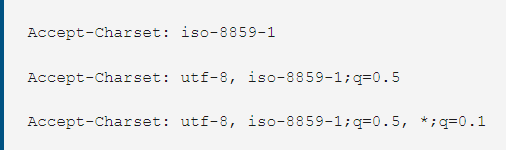
**Date –** основной HTTP заголовок содержащий дату и время, в которое сообщение было создано.

***Заголовки запроса (Request headers*)** – используются в запросах клиента.

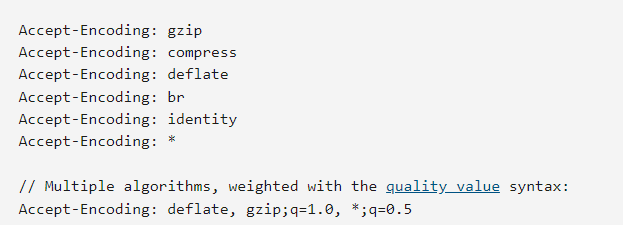
HTTP заголовок запроса **Accept** указывает, какие типы контента, выраженные как MIME типы, клиент может понять.



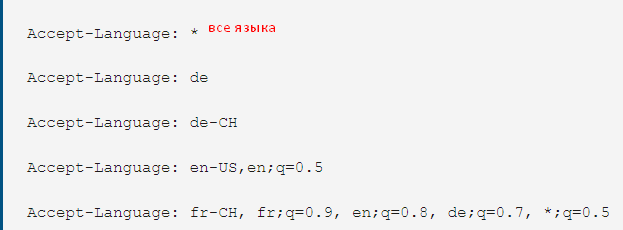
Заголовок **Accept-Charset** запроса HTTP сообщает какую кодировку клиент может понять.



HTTP-заголовок запроса **Accept-Encoding** указывает кодировку содержимого (обычно алгоритм сжатия), которую может понять клиент.

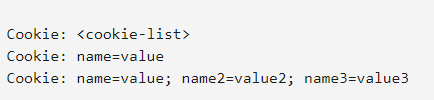


**Accept-Language** сообщает серверу, какие языки клиент понимает и какая локаль предпочтительнее



заголовок **Cookie** содержит сохраненные HTTP-куки, связанные с сервером (т.е. ранее отправленные сервером с помощью заголовка Set-Cookie или установленные в Javascript с помощью Document.cookie).

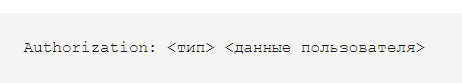
Заголовок Cookie является необязательным и может быть опущен, если, например, настройки конфиденциальности браузера блокируют cookies.



Заголовок **Host** содержит имя домена, для которого предназначен запрос и, опционально, номер порта.

HTTP-запрос отправляется на определенные IP-адреса. Но так как большинство серверов способны размещать несколько сайтов под одним IP, они должны знать, какое доменное имя ищет браузер.

Заголовок HTTP запроса **Authorization** включает в себя данные пользователя для проверки подлинности пользовательского агента.



**<данные пользователя>**

Если используется схема авторизации «Базовая», данные пользователя формируются следующим образом:

* Логин и пароль, разделённые двоеточием (aladdin:opensesame).
* Результирующая строка, закодированная в [base64](https://developer.mozilla.org/en-US/docs/Glossary/Base64) (YWxhZGRpbjpvcGVuc2VzYW1l).

Заголовок **If-Modified-Since** делает запрос условным: сервер отправит обратно запрошенный ресурс со статусом 200, только если он был изменен после указанной даты. Если ресурс не был изменен, ответ будет 304, без тела.

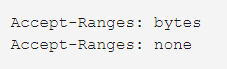


Заголовок HTTP-запроса **If-Range** делает запрос диапазона условным: если условие выполнено, запрос диапазона выдается, и сервер отправляет ответ 206 Partial Content с соответствующим телом. Если условие не выполняется, то полный ресурс отправляется обратно со статусом 200 OK.

**User-Agent -** это строка с характеристиками, по которым сервера и сетевые узлы могут определить тип приложения, операционную систему, производителя и/или версию пользовательского агента(браузера).

***Заголовки ответа (Response headers*)** – используются в ответах серверов.

Заголовок ответа **Accept-Ranges** -- это маркер, который использует сервер, чтобы уведомить клиента о поддержке "запросов по кускам".



Заголовок **Allow** перечисляет набор методов, поддерживаемых ресурсом.

Этот заголовок должен быть отправлен, если сервер отвечает кодом состояния 405 Method Not Allowed, чтобы указать, какие методы запроса могут быть использованы.



**Last-Modified –** дата и время последнего изменения ресурса.

Заголовок ответа **Location** указывает URL, на который следует перенаправить страницу. Он имеет значение только тогда, когда обслуживается с ответом статуса 3xx (перенаправление) или 201 (создано).

В случае создания ресурса он указывает URL-адрес вновь созданного ресурса.

Если код 303, то для перенаправления совершается еще один запрос, используя при этом метод GET.

При других кодах состояния, обычное используется тот же метод, что и при первом запросе.

**Set-Cookie** Когда веб-сервер хочет установить или обновить файл cookie в вашем браузере, он будет использовать этот header.

Каждый файл cookie отправляется как отдельный header. Обратите внимание, что файлы cookie, установленные с помощью JavaScript, не проходят через HTTP headers.

Если дата истечения срока действия не указана, cookie удаляется, когда окно браузера закрыто.

Заголовок **Server** описывает программное обеспечение, используемое сервером, который обработал запрос - то есть, сервер, сгенерировавший ответ.

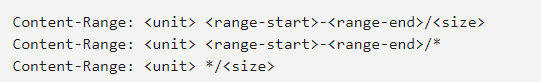
***Заголовки сущности (Entity Headers)***содержут информацию, относящуюся к телу сообщения.

**Content-Encoding –** заголовок сущности, указывающий кодировки примененные для сжатия тела сообщения.

**Content-Language –** указывает язык контента (тела, отправленного получателю).

Заголовок **Content-Length** указывает размер отправленного получателю тела объекта в байтах.

HTTP-заголовок **Content-Range** ответа указывает, к какому месту в полном тексте сообщения относится частичное сообщение (когда получаем ресурс по частям).

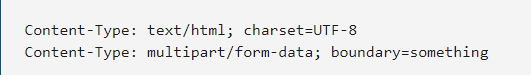




**Content-Type** используется для указания типа ресурса.

В ответах заголовок Content-Type предоставляет клиенту фактический тип содержимого возвращаемого содержимого.

В запросах (таких как POST или PUT) клиент сообщает серверу, какой тип данных будет отправлен.



**Тело**

Тело содержит ресурс, отправляемый на сервер, либо получаемый с сервера.

Включается тело в сообщение или нет зависит как от метода запроса, так и от кода состояния ответа.

**HTTP методы**

Указывают действие, которое нужно выполнить для указанного ресурса.

Метод HTTP является **Безопасным,** если он не меняет состояние сервера. Безопасный метод проводит “только чтение”. GET, HEAD, OPTIONS.

Правильная реализация безопасного метода - это ответственность серверного приложения.

Метод HTTP является **идемпотентным**, если предполагаемый эффект от нескольких идентичных запросов такой же, как и для одного запроса. Корректно реализованные методы GET, HEAD, OPTIONS, PUT и DELETE идемпотентны. Идемпотентность метода не гарантируется сервером, и некоторые приложения могут нарушать ограничение идемпотентности.

**Кешируемые** ответы – это HTTP-ответы, которые могут быть сохранены для дальнейшего восстановления и использованы позже, тем самым снижая число запросов серверу.

**GET –** запрашивает представление указанного ресурса.

GET может быть условным, если он использует заголовки If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, или If-Range.

Частичный GET позволяет запросить указанный фрагмент ресурса. Используется для докачки файлов и быстрого параллельного скачивания в нескольких потоках. Для этого используется заголовок range, в котором указываются байтовые диапазоны. Если сервер не поддерживает частичные запросы, то вернет все содержимое со статусом 200. Иначе возвращает 206(Partial Content), включая заголовок Content-Range.

Запрос GET передает данные в URL в виде пар “имя=значение”. Следуют после знака ? и разделяются знаком &. Например, можно использовать метод GET в HTML форме фильтра товаров: когда нужно, исходя из данных введенных пользователем, переправить его на страницу с отфильтрованными товарами, соответствующими его выбору.

Передавать данные методом GET небезопасно. Отсутствует защита конфиденциальной информации. Введенный пароль будет виден в адресной строке браузера, будет сохранен в истории посещенных сайтов.

Длина URL в методе GET 2048 символов.

Теоретически через URL можно передать файл, закодированный в base64.

**HEAD** – запрашивает заголовки, идентичные тем, что возвращаются после GET-запроса. Может быть выполнен перед загрузкой большого ресурса, чтобы проверить устарел ли ресурс, или можно использовать кэш.

**POST** применяется для передачи пользовательских данных заданному ресурсу.

Метод POST имеет большую степень защиты данных, чем GET: параметры запроса не видны пользователю без использования специального ПО, что дает методу преимущество при пересылке конфиденциальных данных, например в формах авторизации.

Длина URL не ограничена. Примечание: ограничения могут быть установлены сервером.

Бинарные файлы передаются в кодировке base64 - стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит текстово-цифровые латинские символы A-Z, a-z и 0-9 (62 знака) и 2 дополнительных символа, зависящих от системы реализации.

**PUT** создает новый ресурс или заменяет представление целевого ресурса, данными представленными в теле запроса. Предполагает что указанный ресурс существует (что загружаемое содержимое соответствует находящемуся по данному URI ресурсу).

**DELETE** удаляет указанный ресурс.

**OPTIONS** используется для описания параметров соединения с целевым ресурсом. Можно узнать какие методы запросов поддерживаются сервером (в ответе будет запрос Allow).

**PATCH** частично изменяет ресурс.

**CONNECT –** преобразует соединение запроса в прозрачный TCP/IP туннель, обычно, чтобы содействовать установлению защищенного SSL-соединения через нешифрованный прокси.

**TRACE** используется для трассировки, позволяет увидеть что происходит на промежуточных звеньях между клиентом и сервером. TRACE используется при тестировании и отсылает полное сообщение, полученное веб-сервером, обратно клиенту, что позволяет увидеть конкретное содержимое, полученное веб-сервером. Можно посмотреть, что добавили промежуточные сервера.

**Коды ответа HTTP**

Указывают на результат выполнения запроса.

1. 1хх – информационные
2. 2хх – успешные
3. 3хх – перенаправления
4. 4хх – ошибки клиента
5. 5хх – ошибки сервера

100 Continue – продолжить. Запрос успешно принят и клиент может продолжать присылать запросы.

102 Processing – в обработке. Этот код указывает, что сервер получил запрос и обрабатывает его, но обработка еще не завершена.

103 Early Hints - "Ранние подсказки". В ответе сообщаются ресурсы, которые могут быть загружены заранее, пока сервер будет подготавливать основной ответ.

200 OK – Запрос успешно обработан. GET: запрошенный ресурс был найден, и передан в теле ответа. HEAD: заголовки переданы в ответе. POST: ресурс, описывающий результат действия сервера на запрос, передан в теле ответа. TRACE: тело ответа содержит тело запроса полученного сервером.

201 Created – Запрос успешно выполнен и в результате был создан ресурс.

202 Accepted – “Принято”. Запрос принят, но еще не обработан. для случаев, когда запрос обрабатывается другим процессом или сервером, либо для пакетной обработки.

203 Non-Authoritative Information - "Информация не авторитетна". Этот код ответа означает, что информация, которая возвращена, была предоставлена не от исходного сервера, а из какого-нибудь другого источника.

204 No Content - Нет содержимого для ответа на запрос, но заголовки ответа, которые могут быть полезны, присылаются. Клиент может использовать их для обновления кешированных заголовков, полученных ранее для этого ресурса.

206 Partial Content - "Частичное содержимое". Этот код ответа используется, когда клиент присылает заголовок диапазона, чтобы выполнить загрузку отдельно, в несколько потоков.

300 Multiple Choice - Этот код ответа присылается, когда запрос имеет более чем один из возможных ответов. И User-agent или пользователь должен выбрать один из ответов. Не существует стандартизированного способа выбора одного из полученных ответов.

301 Moved Permanently - "Перемещён на постоянной основе". Этот код ответа значит, что URI запрашиваемого ресурса был изменён. Возможно, новый URI будет предоставлен в ответе.

302 Found - Этот код ответа значит, что запрошенный ресурс *временно изменён*. Новые изменения в URI могут быть доступны в будущем. Таким образом, этот URI, должен быть использован клиентом в будущих запросах.

303 See Other - Этот код ответа присылается, чтобы направлять клиента для получения запрашиваемого ресурса в другой URI с запросом GET.

304 Not Modified - Используется для кеширования. Это код ответа значит, что запрошенный ресурс не был изменён. Таким образом, клиент может продолжать использовать кешированную версию ответа.

307 Temporary Redirect - "Временное перенаправление". Сервер отправил этот ответ, чтобы клиент получил запрошенный ресурс на другой URL-адрес с тем же методом, который использовал предыдущий запрос. Данный код имеет ту же семантику, что код ответа 302 Found, за исключением того, что агент пользователя не должен изменять используемый метод HTTP.

400 Bad Request - Этот ответ означает, что сервер не понимает запрос из-за неверного синтаксиса.

401 Unauthorized - Для получения запрашиваемого ответа нужна аутентификация. Статус похож на статус 403, но,в этом случае, аутентификация возможна.

403 Forbidden - "Запрещено". У клиента нет прав доступа к содержимому, поэтому сервер отказывается дать надлежащий ответ.

404 Not Found - Сервер не может найти запрашиваемый ресурс.

405 Method Not Allowed - "Метод не разрешён". Сервер знает о запрашиваемом методе, но он был деактивирован и не может быть использован. Два обязательных метода,  GET и HEAD,  никогда не должны быть деактивированы и не должны возвращать этот код ошибки.

408 Request Timeout - сервер хотел бы отключить это неиспользуемое соединение.

409 Conflict - Этот ответ отсылается, когда запрос конфликтует с текущим состоянием сервера.

411 Length Required - Запрос отклонён, потому что сервер требует указание заголовка Content-Length, но он не указан.

412 Precondition Failed - Клиент указал в своих заголовках условия, которые сервер не может выполнить.

416 Requested Range Not Satisfiable - Диапазон указанный заголовком запроса Range не может быть выполнен;

500 Internal Server Error - "Внутренняя ошибка сервера". Сервер столкнулся с ситуацией, которую он не знает как обработать.

501 Not Implemented - Метод запроса не поддерживается сервером и не может быть обработан. Единственные методы, которые сервера должны поддерживать (и, соответственно, не должны возвращать этот код) -  GET и HEAD.

503 Service Unavailable - Сервер не готов обрабатывать запрос. Зачастую причинами являются отключение сервера или то, что он перегружен. Обратите внимание, что вместе с этим ответом удобная для пользователей(user-friendly) страница должна отправлять объяснение проблемы.

504 Gateway Timeout - Этот ответ об ошибке предоставляется, когда сервер действует как шлюз и не может получить ответ вовремя. (Сетевой шлюз (Gateway)— это маршрутизатор или какое-либо программное обеспечение, которое позволяет двум и более независимым сетям с разными протоколами обмениваться между собой данными).

505 HTTP Version Not Supported - HTTP-версия, используемая в запросе, не поддерживается сервером.

**Управление кешированием**

Осуществляется с помощью заголовка cache-control.

Так же ответ сервера может содержать заголовок Last-Modified, в котором указано время последнего обновления ресурса. При последующем запросе, браузер может отправить заголовок If-Modified-Since с указанием времени. Если ресурс был изменен, то вернется обновленная версия. Если ресурс не был изменен, возвращается код 304 Not Modified, который указывает, что нет необходимости повторно передавать запрошенные ресурсы. Это неявное перенаправление на кешированный ресурс.

Веб сервер может отправлять заголовок ETag, который является идентификатором специфичной версии ресурса. Браузер сохраняет это значение, так как он кеширует запрос. В следующий раз, когда браузер запрашивает тот же ресурс, он добавляет заголовок If-None-Match с Etag. Если Etag совпадают, то сервер отправляет код 304, и никакого содержимого. Браузер будет загружать содержимое из кеша.

Заголовок HTTP-ответа Vary определяет, как по заголовкам будущих запросов понять, может ли быть использована копия из кеша, или нужно запросить новые данные у сервера.

**Cookie**

Cookie – это небольшой фрагмент данных, отправляемый сервером, который клиент может сохранить и отсылать обратно с новым запросом к данному серверу. Они запоминают информацию о состоянии для протокола HTTP.

Cookie используются, главным образом, для:

⦁    Управления сеансом (логины, аутентификация, корзины для виртуальных покупок)  
⦁    Персонализации (пользовательские предпочтения)  
⦁    Мониторинга (отслеживания поведения пользователя)

Сервер может отправить заголовок Set-Cookie с ответом. Cookie обычно запоминаются браузером и посылаются в значении заголовка Cookie с каждым новом запросом к одному и тому же серверу. Можно задать срок действия cookie, а также срок его жизни, после которого cookie не будет отправляться. Также можно указать ограничения на путь и домен.

Если для Cookie не заданы атрибуты Expires и Max-Age то куки называется *сессионным*. Такие куки удаляются при закрытии клиента (если в браузере не включено восстановление сеанса).

*Постоянные куки* удаляются при наступлении определенной даты (атрибут Expires) или после определенного интервала времени (Max-Age).

Не следует хранить важные данные в cookie. Их механизм уязвим, а флаг secure никого шифрования не обеспечивает.

Куки HTTPonly не доступны из JavaScript через свойства Document.cookie.

Атрибут Domain указывает хосты, на которые отсылаются куки. По умолчанию берется доменная часть адреса документа. Атрибут Path указывает URL, который должен быть в запрашиваемом ресурсе на момент отправки заголовка cookie.

**Redirect**

Redirect – автоматическая переадресация пользователя с одного адреса страницы на другой. Вызывается при отправке сервером ответа с кодом 3хх. Когда браузер получает ответ перенаправления, он делает другой запрос, на новый предоставленный URL.

Виды:

* Постоянные – подразумевают что оригинальный URL больше не должен использоваться. 301, 308.
* Временные 302, 303, 307
* Not Modified 304, перенаправление на локальную закешированную копию.
* Multiple Choice 303, указывает, что запрос имеет несколько возможных ответов. Пользователь-агент или пользователь должны выбрать один из них.

**CORS**

Cross-Origin Resource Sharing – механизм, использующий HTTP-заголовки, чтобы дать возможность клиенту получать разрешения на доступ к ресурсам другого домена.

Сервер с помощью заголовков описывает набор источников, которым разрешено читать информацию, запрашиваемую веб браузером, а также должны ли отправляться Cookies и HTTP Authentification данные.

Есть 3 категории доступа:

1. Запись из разных источников – ссылки, переадресации и отправка форм.
2. Вставка из разных источников – теги, загружаемые через <script>, <link>, <img>, <video>, <audio>, <object>, <embed>, <iframe> и т.п. Все они разрешены по умолчанию.
3. Считывание из разных источников — это теги, загружаемые через вызовы AJAX/ fetch. Все они по умолчанию заблокированы вашим браузером.

При простых кросдоменных запросах браузер ожидает заголовок Access-Control-Allow-Origin: <origin> | \* - указывает браузеру разрешить источнику origin доступ к ресурсу, \* разрешает запросы из любых источников. При этом в запросе отправляется заголовок Origin, с адресом ресурса, откуда пришел запрос.

Для того чтобы использовать куки (или другие учетные данные) в кроссдоменном запросе, необходимо чтобы в ответе присутсвовал заголовок Access-Control-Allow-Credentials: True. При этом в заголовке Control-Allow-Origin не должна быть \*.

Предварительные запросы OPTIONS выполняются перед запросами, которые cors считает сложными. Они спрашивают разрешения, можно ли выполнить запрос с определенными методами и заголовками. Этот запрос может отправлять заголовки Access-Control-Request-Method (метод которым будем обращаться), Access-Control-Request-Headers (заголовки которые отправим с запросом).

В ответе приходят следующие заголовки:

* Access-Control-Allow-Methods – методы, с которыми можно обращаться к ресурсу.
* Access-Control-Allow-Headers – заголовки, которые можно включать в запрос.
* Control-Allow-Origin – каким доменам разрешено обращаться к ресурсу.
* Access-Control-Max-Age, который указывает число секунд (5 по умолчанию) и это значение соответствует периоду, на который предоставляемая заголовками Access-Control-Allow-Methods и Access-Control-Allow-Headers информация может быть кэширована.

**Разница между версиями HTTP**

HTTP/1.0 открывал TCP-соединение для каждого обмена запросом/ответом.