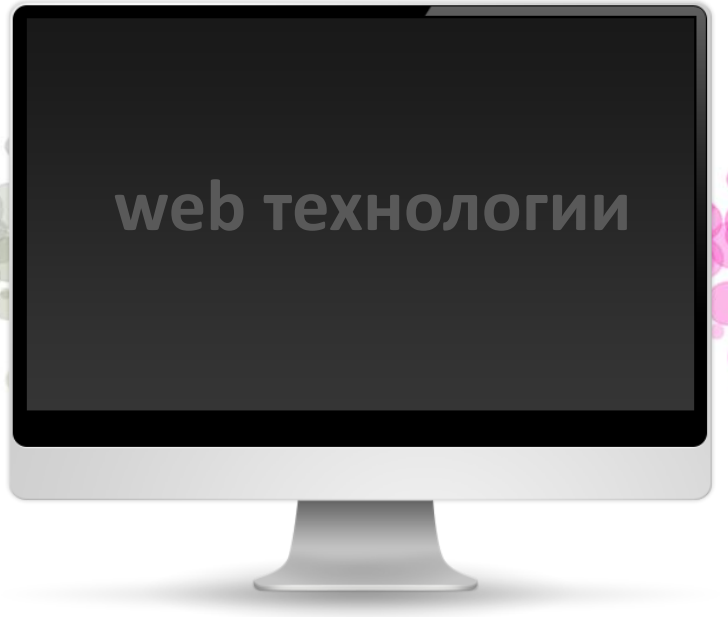
Брестский государственный
технический университет



HTTP

ФАКУЛЬТЕТ
ЭЛЕКТРОННО-
ИНФОРМАЦИОННЫХ
СИСТЕМ

Весна 2022

План лекции

- Как работает сеть?
- Что такое HTTP?
- Основы HTTPS
- Основные инструменты HTTP

Что происходит когда вы просматриваете веб-страницы?

Что же там происходит?

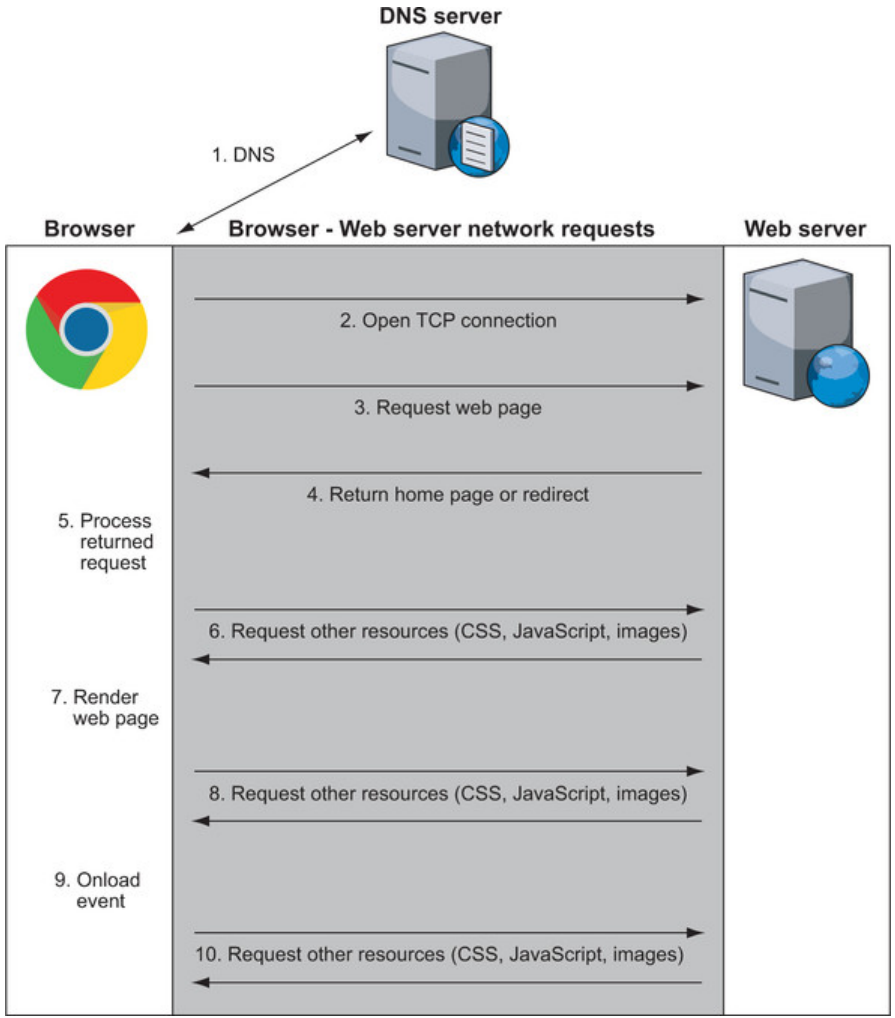
Предположим, что вы запускаете браузер и переходите на сайт **www.google.com.**

1. Браузер запросит реальный адрес www.google.com с сервера системы доменных имен (Domain Name System, DNS), который переведет понятное человеку имя www.google.com в удобный для машины IP-адрес.

Что же там происходит?

Этот IP-адрес будет являться либо адресом более старого формата IPv4, который относительно понятен человеку (например, 216.58.192.4)

либо адресом нового формата IPv6, который могут обрабатывать только машины (например, 2607: f8b0:4005:801:0:0:0:2004).


DNS server
1. DNS
Browser
Browser - Web server network requests
Web server
2. Open TCP connection
3. Request web page
4. Return home page or redirect
5. Process returned request
6. Request other resources (CSS, JavaScript, images)
7. Render web page
8. Request other resources (CSS, JavaScript, images)
9. Onload event
10. Request other resources (CSS, JavaScript, images)

Что же там происходит?

По причине глобального характера сети Internet крупные компании часто имеют несколько серверов по всему миру.

Когда вы запрашиваете IP-адрес у DNS, он обычно предоставляет IP-адрес ближайшего сервера, благодаря чему вы получаете наиболее быстрый доступ к веб-страницам. Например, ответ на запрос **IP-адреса для www.google.com в Америке и в Европе будет отличаться.**

Где же версия IPv5?

Если Internet-протокол версии 4 (IPv4) был заменен версией 6 (IPv6), то где же тогда версия 5?

И почему вы никогда не слышали об IPv1 или IPv3?

Где же версия IPv5?

Первые четыре бита в IP-пакете используются для хранения версии протокола.

В теории максимально возможная версия – это версия 15.

До того как IPv4 начали использовать повсеместно, существовало 4 экспериментальных версии:

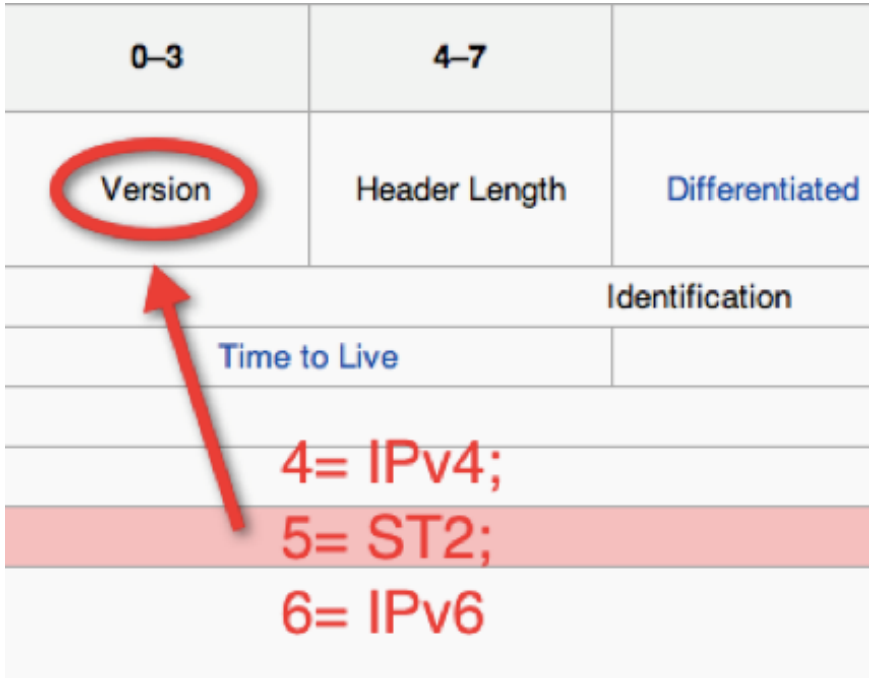
(версия 0 – версия 3).

Где же версия IPv5?

Однако ни для одной из них, кроме версии 4, не было создано официального стандарта.

Версия 5 предназначалась для протокола реального времени Internet Stream Protocol, обеспечивающего потоковую передачу аудио и видео в реальном времени, подобно современной IP-телефонии (Voice over IP, VoIP).

Где же версия IPv5?



Где же версия IPv5?

Однако версия IPv5 так и не стала популярной, так как имела те же ограничения адресного пространства, что и версия 4.

Когда появилась версия 6, работа над IPv5 была остановлена, и версия 6 стала преемником IPv4.

Где же версия IPv5?

Изначально IPv6 была названа версией 7, потому что многие ошибочно полагали, что версия 6 уже использована.

Номера 7, 8 и 9 также применялись для нумерации версий, но эти версии больше не используются.

Если когда-нибудь и появится преемник IPv6, то это, скорее всего, будет IPv10 или более поздняя версия.

Что же там происходит?

2. Через веб-браузер компьютер устанавливает TCP-соединение по IP-адресу через стандартный сетевой порт (порт 80) или стандартный защищенный сетевой порт (порт 443).

Передача информационных потоков в Internet осуществляется с помощью протокола IP.

Что же там происходит?

Протокол TCP обеспечивает надежность при передаче данных, а также возможность повторной передачи («Здравствуйте, вы все поняли?», «Нет, не могли бы вы повторить последний бит, пожалуйста?»).

Эти две технологии часто используются вместе, поэтому их называют TCP/IP.

Именно на основе этих протоколов создана большая часть сервисов в Internet.

Что же там происходит?

3. После того как браузер подключился к веб-серверу, он начинает отправлять запросы веб-сайту.

На этом этапе в дело вступает протокол HTTP. Однако мы рассмотрим порядок его работы чуть позже.

Что же там происходит?

4. Сервер Google отвечает вне зависимости от типа URL-адреса, который вы запрашиваете. Как правило, ответ приходит в виде текста веб-страницы в формате HTML.

Что же там происходит?

HTML – это стандартизированная, структурированная система разметки текстового содержимого веб-страниц.

Документ на языке HTML обычно представляет собой набор элементов, начало и конец которых определяется HTML- тегами и ссылками на блоки другой информации, необходимой для создания мультимедийных веб-страниц, которые вы привыкли видеть (каскадные таблицы стилей [CSS], код JavaScript, изображения, шрифты и т. д.).

Что же там происходит?

Однако иногда вместо HTML-страницы ответом может быть перенаправление на верный адрес.

Например, Google работает только на HTTPS, поэтому, если вы запросите URL

http://www.google.com, ответом будет HTTP-инструкция (обычно код такого ответа 301 или 302), которая перенаправит вас на новый адрес https://www.google.com.

Что же там происходит?

Такой ответ запускает процессы некоторых или всех предыдущих шагов снова, в зависимости от того, как выглядит адрес перенаправления.

Он может иметь другой порт на другом сервере, другой порт на том же сервере (например, перенаправление на HTTPS) или это может быть другая страница на том же сервере и порте.

Что же там происходит?

5. Затем браузер обрабатывает возвращенный запрос. Браузер рассчитан на ответы в формате HTML, поэтому он начинает анализировать HTML-код и строит в памяти объектную модель документа (**document object model, DOM**), которая отображает внутреннее строение страницы. Скорее всего, в процессе анализа веб-браузер найдет и другие ресурсы, необходимые для правильного отображения страницы (например, CSS, JavaScript и изображения).

Что же там происходит?

6. Веб-браузер запросит необходимые ему дополнительные ресурсы. Веб-страницы Google используют небольшое количество ресурсов. На текущий момент их требуется всего 16.

Каждый из этих ресурсов запрашивается аналогичным образом, следуя шагам 1–6, так как эти ресурсы в свою очередь могут запрашивать другие ресурсы.

Что же там происходит?

Как правило, страницы обычных веб-сайтов более разнообразны, чем страницы Google.

Такие веб-сайты нуждаются в 75 ресурсах и зачастую из разных доменов, поэтому шаги 1–6 должны быть выполнены для каждого ресурса. Именно из-за таких ситуаций и замедляется работа браузера.

Что же там происходит?

7. Когда браузер получает достаточно критически необходимых ресурсов, он начинает отображать страницу на экране.

Старт рендеринга веб-страницы является не таким простым процессом, как кажется.

Что же там происходит?

Если веб-браузер начнет отображать страницу слишком рано, она будет «прыгать» по мере загрузки большего количества контента.

Особенно раздражает, когда страница начинает «прыгать» после того, как вы прочитали уже половину статьи.

Что же там происходит?

8. После отображения страницы веб-браузер продолжает в фоновом режиме загружать другие ресурсы, необходимые странице, и обновляет страницу по мере их обработки.

Что же там происходит?

Он загружает второстепенные элементы, такие как изображения и скрипты отслеживания рекламы.

Именно поэтому бывает так, что, когда вы загружаете веб-страницу, на ней какое-то время не отображаются изображения (особенно при медленном соединении), а затем, по мере их загрузки, они появляются.

Что же там происходит?

9. Когда страница полностью загружена, браузер останавливает значок загрузки (в большинстве браузеров вращающийся значок в адресной строке или рядом с ней) и запускает событие OnLoad Java Script, которое является маркером для JavaScript и означает, что страница готова к работе.

Что же там происходит?

10. Времена, когда веб-страница была статична, уже давно прошли, поэтому браузер продолжает отправлять запросы, даже когда она полностью загружена.

Сегодня многие веб-страницы являются многофункциональными приложениями, взаимодействующими с различными серверами для отправки или загрузки дополнительного контента. Отображение контента может быть инициировано действиями пользователя.

Что же там происходит?

Например, вы вводите запросы в строку поиска на домашней странице Google и мгновенно видите предложенные варианты запроса без нажатия кнопки поиска.

Также это могут быть действия, управляемые приложением, например автоматическое обновление ленты Facebook или Twitter без нажатия кнопки обновления.

Что же там происходит?

Эти действия часто происходят в фоновом режиме.

Они скрыты от вас.

В качестве примера можно привести рекламные и аналитические скрипты, которые отслеживают ваши действия на сайте, для того чтобы сообщать аналитические данные владельцам веб-сайтов и/или рекламным сетям.

Что такое HTTP?

Как уже было сказано, HTTP расшифровывается как «протокол передачи гипертекста».

Как следует из его названия, HTTP изначально предназначался исключительно для передачи гипертекстовых документов (документов, содержащих ссылки на другие документы). Первая версия не могла передать ничего, кроме таких документов.

Что такое HTTP?

Название «протокол передачи гипертекста» не совсем актуально, так как разработчики быстро поняли, что протокол может быть использован для передачи других типов файлов (например, изображений). Однако, сейчас HTTP настолько распространен, что переименовывать его уже слишком поздно.

Что такое HTTP?

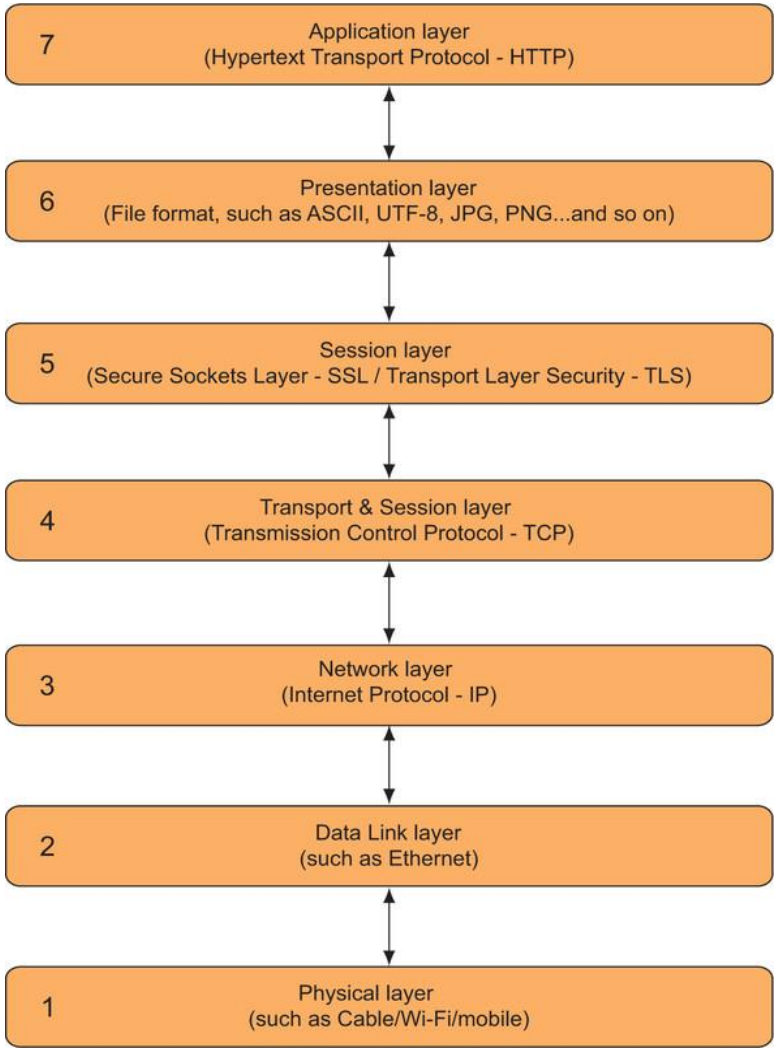
Модель взаимодействия открытых систем (Open Systems Interconnection, OSI) – это многоуровневая модель сетевых протоколов.

Она состоит из 7 уровней, хотя они соотносятся с сетями, и особенно с Internet-трафиком, с некоторой долей условности.

Что такое HTTP?

TCP (Transmission Control Protocol) является рабочим протоколом как минимум двух уровней, а возможно, и трех (это зависит от того, как вы определяете эти уровни).

На рисунке можно увидеть, как данная модель соотносится с Internet-трафиком и как в нее вписывается протокол HTTP.


7
Application layer
(Hypertext Transport Protocol - HTTP)
6
Presentation layer
(File format, such as ASCII, UTF-8, JPG, PNG...and so on)
5
Session layer
(Secure Sockets Layer - SSL / Transport Layer Security - TLS)
4
Transport & Session layer
(Transmission Control Protocol - TCP)
3
Network layer
(Internet Protocol - IP)
2
Data Link layer
(such as Ethernet)
1
Physical layer
(such as Cable/Wi-Fi/mobile)

Что такое HTTP?

До сих пор ведутся споры о точном определении уровней в модели.

В Internet, как и в других сложных системах, не все поддается классификации, и разграничить что-либо бывает не так легко, как этого хотелось бы разработчикам.

Согласно мнению Инженерного совета интернета, не обязательно делать слишком большой упор на деление модели по уровням.

Что такое HTTP?

Однако на более высоком уровне такой подход может помочь понять, как в модель вписывается HTTP, а также как она зависит от других протоколов.

Многие веб-приложения используют HTTP, поэтому в таких случаях прикладной уровень может больше относиться к сетевому уровню, чем к приложениям JavaScript.

Что такое HTTP?

По сути, HTTP является протоколом запроса и ответа.

Браузер совершает запрос к веб-серверу, используя синтаксис HTTP.

Веб-сервер в свою очередь отвечает сообщением, содержащим запрошенный ресурс. HTTP широко распространен именно благодаря своей простоте.

Что такое HTTP?

После того как открывается соединение, базовый синтаксис HTTP-запроса выглядит следующим образом:

GET /page.html↵ ,

где символ ↵ означает возврат каретки / начало новой строки.

HTTP/0.9

В 1991 году была опубликована первая спецификация 2 протокола HTTP версии 0.9. Она изложена в краткой форме и состоит менее чем из 700 слов.

HTTP/0.9

Согласно этой спецификации при использовании HTTP 0.9 соединение с сервером и дополнительным портом (если порт не указан, то по умолчанию происходит соединение с портом 80) осуществляется по протоколу TCP/IP (или с помощью аналогичной службы, ориентированной на установку предварительного соединения).

HTTP/0.9

Для этого следует сделать запрос в виде всего одной строки ASCII-текста, состоящей из команды GET, адреса документа (без пробелов) и символов возврата каретки и перевода строки (возврат каретки необязателен).

Сервер должен ответить сообщением в формате HTML, которое в спецификации описано как «байтовый поток символов ASCII».

HTTP/0.9

После каждого запроса соединение закрывается сервером (как это было показано в предыдущих примерах).

Также, согласно спецификации, «ответы на ошибки предоставляются в виде понятного человеку текста в формате HTML».

HTTP/0.9

Отличить ответ об ошибке от корректного ответа можно только по содержимому текста, иного способа не существует.

Ответ заканчивается так: «Запросы идемпотентны. Сервер не сохранит данные о запросе при отключении от сети».

HTTP/0.9

HTTP 0.9 не фиксирует данные о запросах, поэтому с одной стороны он является благословлением (так как он очень прост), а с другой – проклятием (так как для создания сложных приложений необходимо использовать другие технологии, например HTTP-куки).

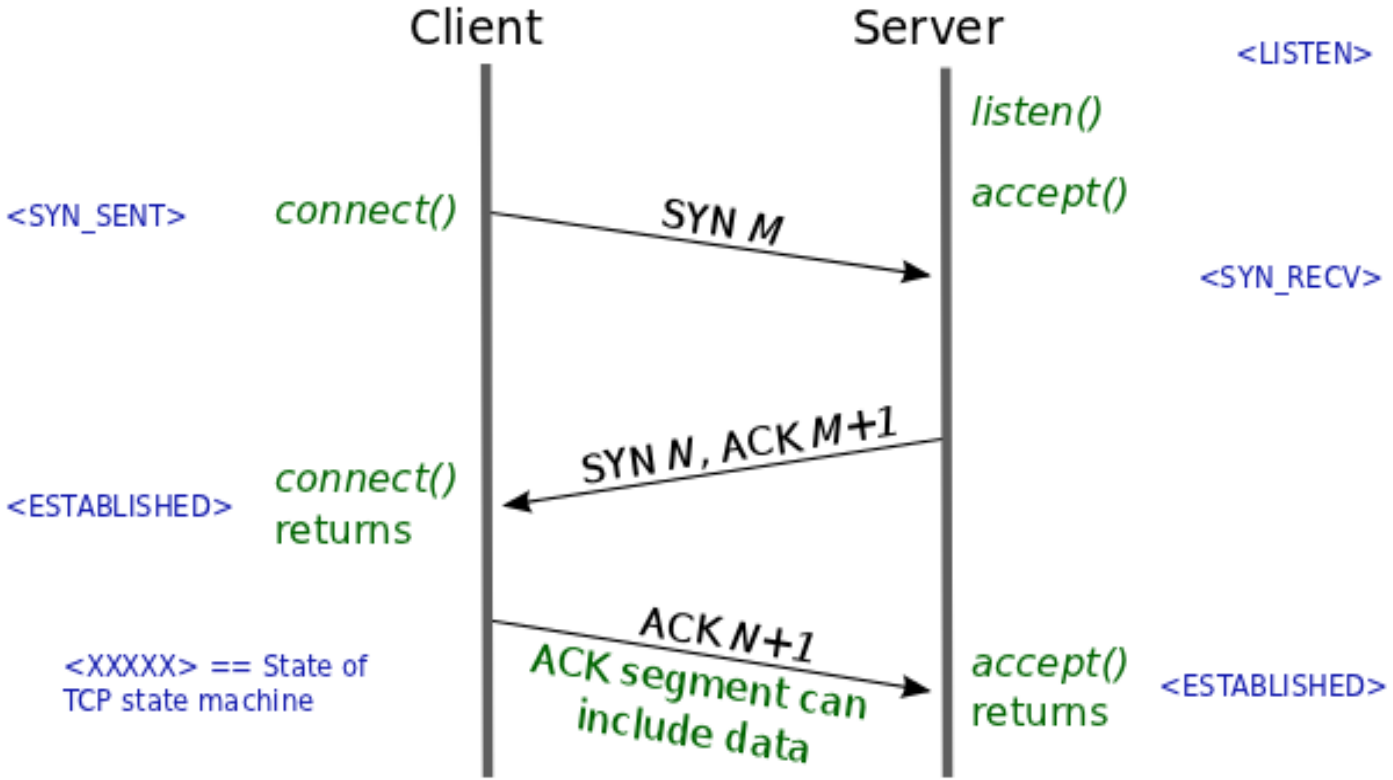
HTTP/0.9

На этом слайде приведена единственная возможная команда для HTTP/0.9:

GET /section/page.html↵

Синтаксис носит фиксированный характер, за исключением, конечно, запрашиваемого ресурса (/section/page.html). В этой версии еще не существовало концепции полей заголовка, а также она не могла работать с медиафайлами например с изображениями.

HTTP/0.9


Client
Server
<LISTEN>
listen()
accept()
<SYN_SENT>
connect()
SYN M
<SYN_RECV>
SYN N, ACK M+1
<ESTABLISHED>
connect()
returns
ACK N+1
ACK segment can
include data
accept()
returns
<ESTABLISHED>
<XXXXX> == State of
TCP state machine

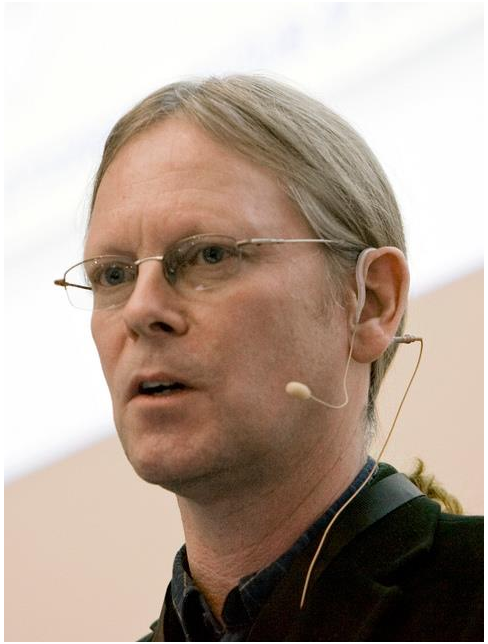
HTTP/1.0

Всемирная паутина возымела почти мгновенный успех. По данным NetCraft, к сентябрю 1995 года в ней насчитывалось уже **19 705** хостов. Спустя месяц их количество возросло до **31 568**.

К 1995 году стало ясно, что функционала простейшего HTTP/0.9 уже недостаточно, а большинство веб-серверов реализовало расширения, выходящие далеко за рамки его спецификации.

HTTP/1.0

Рабочая группа The HTTP Working Group (HTTP WG), возглавляемая Дейвом Раггеттом (Dave Raggett), начала работать над HTTP/1.0 в попытке задокументировать «общее использование протокола».



В мае 1996 года рабочей группой IETF был опубликован документ под названием RFC 1945 1 (Request for Comments – «Рабочее предложение»).

HTTP/1.0

HTTP/1.0 добавил в протокол некоторые ключевые обновления, такие как:

- дополнительные запросы HEAD и POST, помимо уже существующего GET;
- указание номеров HTTP-версии. Изначально предполагалось, что по умолчанию будет использоваться HTTP/0.9 для реализации обратной совместимости;

HTTP/1.0

- HTTP-заголовки, которые использовались как в запросах, так и в ответах и должны были предоставлять больше информации о запрашиваемом ресурсе и отправляемом ответе;
- трехзначный код ответа, который указывал, был ли ответ успешным. Подобные коды свидетельствовали о запросах перенаправления, условных запросах и отображали статус ошибки (например,один из самых известных кодов 404 – Not Found).

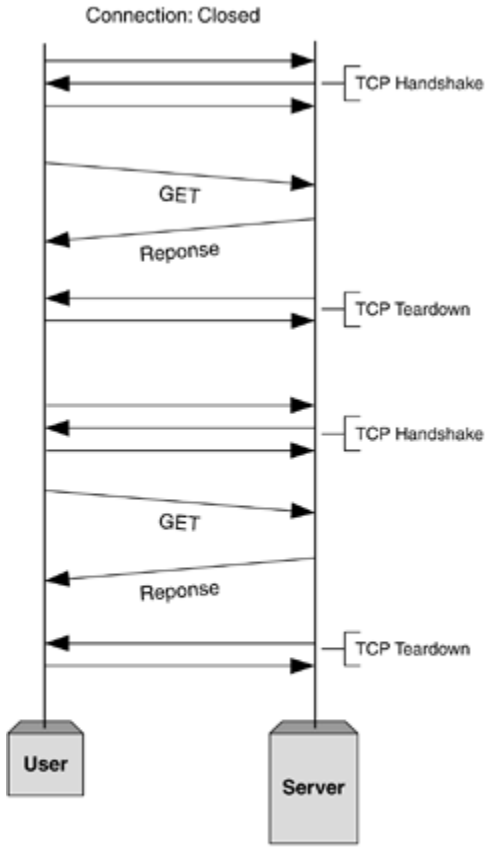
HTTP/1.0

Такие усовершенствования были очень важны, поскольку возникли из реальной необходимости. HTTP/1.0 был создан не столько для внедрения каких-либо новых опций, сколько для того, чтобы задокументировать изменения, уже произошедшие в работе веб-серверов.

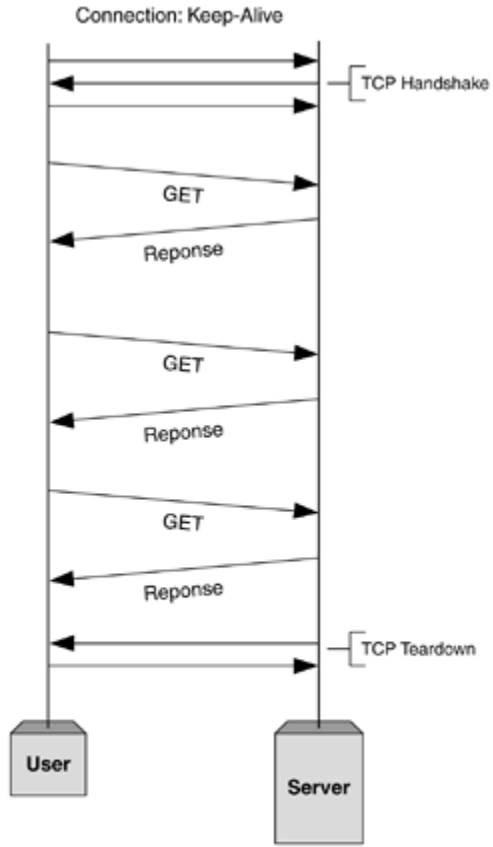
HTTP/1.0

Такие изменения открыли для Internet множество новых возможностей.

Например, пользователи получили возможность добавлять на веб-страницы медиафайлы посредством заголовков HTTP-ответов, позволяющих определить тип содержимого данных в теле страницы.



In a default HTTP/1.0 session, the TCP connection will be torn down and re-established between each HTTP GET request.



In a default HTTP/1.1 session, a single TCP connection will be held, open and multiple GET requests will be passed across.

Методы HTTP/1.0

Метод GET остался почти таким же, как и в HTTP/0.9.

Однако в HTTP/1.0 появились заголовки, что позволило создавать GET-запросы с условием, помощью которых клиент может уточнить, что хочет получить ресурс, только если он изменился с момента последнего получения – если страница не была изменена, клиент получает соответствующий ответ и продолжает использовать ранее полученную копию ресурса.

Методы HTTP/1.0

С появлением метода HEAD клиенты смогли получать все метаданные ресурса (например, HTTP-заголовки), не загружая при этом сам ресурс.

Методы HTTP/1.0

Этот метод полезен по многим причинам. Например, поисковая система, такая как Google, может проверить, был ли ресурс обновлен, и загрузить его уже в обновленном виде, что сэкономит ресурсы для обеих сторон.

Методы HTTP/1.0

Появление метода POST позволило клиенту отправлять данные непосредственно на веб-сервер.

Пользователи могли работать с файлом непосредственно через HTTP (при условии, что веб-сервер настроен на получение данных), вместо того чтобы выгружать новый HTML-файл на сервер, используя стандартные методы передачи данных.

Методы HTTP/1.0

Метод POST может использоваться как при работе с целыми файлами, так и с небольшими фрагментами произвольных данных.

Методы HTTP/1.0

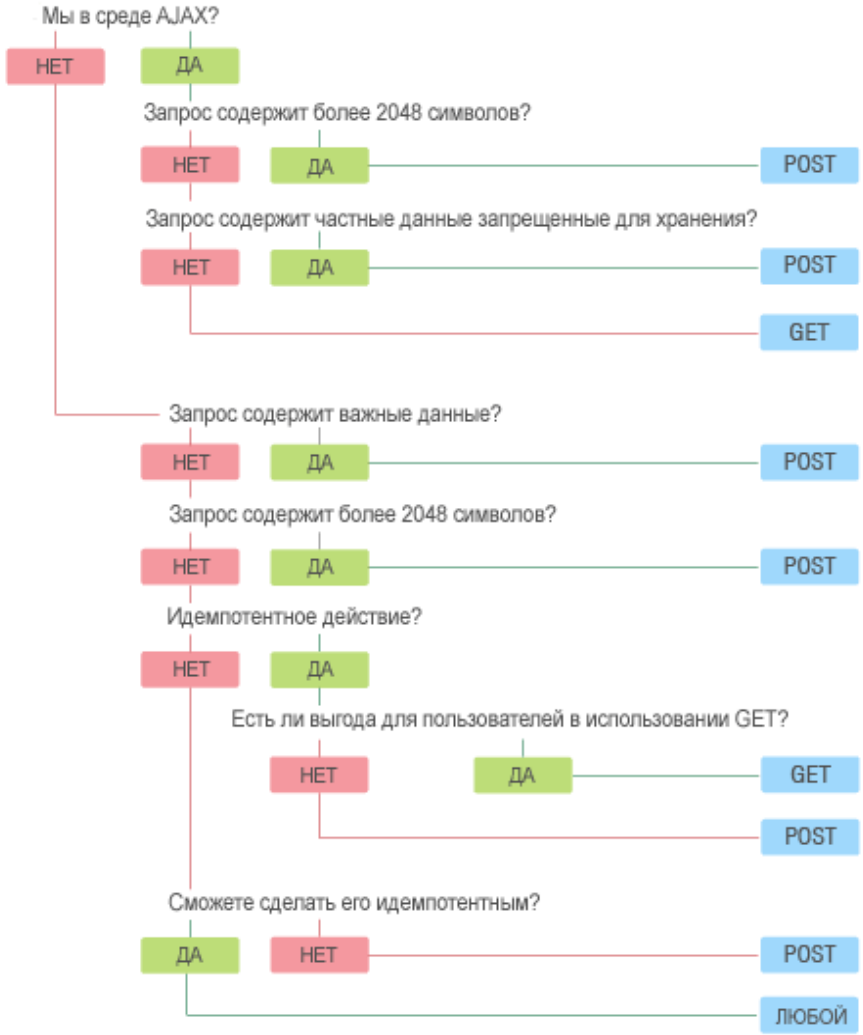
Обычно этот метод используется в веб-формах для ввода данных на различных веб сайтах, где содержимое веб-формы отправляется как пара «поле–значе-ние» в виде HTTP-запроса.

Таким образом, метод POST позволяет клиенту отправлять информацию на сервер в виде HTTP-запроса, имеющего собственное тело, как у HTTP-ответов.

Методы HTTP/1.0

С помощью GET вы можете отправлять данные посредством параметров запроса прямо в URL-адресе, поместив их после знака **«?»**.

Например, запрос в виде https://www.google.com/?q=search+string сообщит поисковой системе Google, что вас интересует search string.


Мы в среде AJAX?
НЕТ
ДА
Запрос содержит более 2048 символов?
НЕТ
ДА
POST
Запрос содержит частные данные запрещенные для хранения?
НЕТ
ДА
POST
GET
Запрос содержит важные данные?
НЕТ
ДА
POST
Запрос содержит более 2048 символов?
НЕТ
ДА
POST
Идемпотентное действие?
НЕТ
ДА
Есть ли выгода для пользователей в использовании GET?
НЕТ
ДА
GET
POST
Сможете сделать его идемпотентным?
ДА
НЕТ
POST
ЛЮБОЙ

Методы HTTP/1.0

URL-адреса ограничены в длине и содержании (например, в них не могут быть использованы двоичные данные). URL-адрес не должен содержать конфиденциальные данные (пароли, данные кредитных карт и т. д.), так как в таком случае их можно будет увидеть на экране и в истории браузера.

Методы HTTP/1.0

Таким образом, метод POST является более безопасным способом отправки данных, так как он обеспечивает конфиденциальность личных данных (тем не менее следует быть осторожным при отправке таких данных по обычному HTTP соединению).

Методы HTTP/1.0

Еще одно отличие заключается в том, что GET-запрос идемпотентен, а POST-запрос таковым не является. Это означает, что при отправке нескольких GET-запросов на один и тот же URL-адрес, ответы на все запросы будут одинаковыми, а при отправке нескольких POST-запросов на один и тот же URL-адрес так случается не всегда.

Методы HTTP/1.0

Например, когда вы обновляете веб-страницу, после обновления она должна остаться в неизменном виде.

А когда вы обновляете страницу с подтверждением оплаты на торговой онлайн-площадке, браузер уточнит: «Вы уверены, что хотите отправить данные повторно? Это может привести к совершению дополнительной покупки»

Заголовки HTTP- запросов

В версии HTTP/0.9 для осуществления GET-запроса была отведена единственная строка, а в версии HTTP/1.0 появились заголовки.

Они позволили через запрос предоставлять серверу дополнительную информацию, которая помогла бы обработать запрос эффективнее.

Заголовки HTTP- запросов

Для HTTP-заголовков предназначены отдельные строки после начальной строки запроса.

Таким образом, HTTP-запрос GET будет выглядеть не как

GET /page.html↲

а как

GET /page.html HTTP/1.0 ↲

Header1: Value1 ↲

Header2: Value2 ↲

Заголовки HTTP- запросов

или без заголовков:

GET /page.html HTTP/1.0 ↲

↲

То есть к начальной строке был добавлен опциональный раздел версии (по умолчанию HTTP/0.9), а за опциональным разделом заголовка HTTP следовали два символа возврата каретки (новой строки, далее для краткости называемые символами возврата) в конце вместо одного.

Заголовки HTTP- запросов

Второй символ возврата был необходим для отправки пустой строки, которая указывала на завершение опционального раздела заголовка запроса.

Заголовки HTTP- запросов

Заголовки HTTP имеют следующий вид: имя заголовка, двоеточие и далее содержимое заголовка. Согласно спецификации имя заголовка (не содержимое) не чувствительно к регистру.

Заголовки HTTP- запросов

Заголовки можно разместить на нескольких строках, но тогда каждую новую строку нужно начинать с пробела или табуляции. Но так делать не рекомендуется, потому что не все клиенты или серверы используют этот формат и поэтому могут обработать заголовки неправильно.

Вместо этого можно отправить несколько заголовков одного типа, которые семантически будут идентичны отправке версий, разделенных запятыми.

Заголовки HTTP- запросов

GET/page.html HTTP/1.0↲
Header1: Value1↲
Header1: Value2↲

обрабатывается также, как и

GET /page.html HTTP/1.0↲
Header1: Value1, Value2↲

Типичный GET-запрос

GET /page.html HTTP/1.0↵

Accept: text/html,application/xhtml+xml,image/jxr/,*/*↵

Accept-Encoding: gzip, deflate, br↵

Accept-Language: en-GB,en-US;q=0.8,en;q=0.6↵

Connection: keep-alive↵

Host: www.example.com↵

User-Agent: MyAwesomeWebBrowser 1.1↵

↵

Типичный ответ от сервера

HTTP/1.0 200 OK

Date: Sun, 25 Jun 2017 13:30:24 GMT

Content-Type: text/html

Server: Apache

<!doctype html>

<html>

<head>

...и т. д.

Коды HTTP- ответов

Остальная часть кода HTML представлена следующим образом. Первая строка ответа включает информацию о HTTP-версии ответного сообщения (HTTP/1.0), трехзначный код состояния HTTP (200) и текстовое описание кода состояния (OK). Коды состояния и описания появились в HTTP/1.0. В HTTP/0.9 такого понятия, как код ответа, не существовало, а ошибки могли встретиться только в возвращаемом HTML документе.

Коды HTTP- ответов

Категория	Код	Наименование	Описание
1xx (информационные)	нет	нет	HTTP/1.0 не определяет никаких кодов состояния 1xx, но определяет категорию
2xx (успешно)	200	OK	Стандартный код ответа для успешного запроса
	201	Created	Код должен быть возвращен по POST-запросу
	202	Accepted	Обработка запроса в процессе
	204	No content	Запрос успешно принят и обработан, но в ответе нет тела сообщения

Категория	Код	Наименование	Описание
3xx (перенаправление)	300	Multiple choices	Данный код используется редко. В нем говорится, что категория 3xx подразумевает, что ресурс доступен в одном (или нескольких) местах, а точный ответ предоставляет более подробную информацию о том, где он находится
	301	Moved permanently	Заголовок Location HTTP-ответа должен предоставить новый URL ресурса
	302	Moved temporarily	Заголовок Location HTTP-ответа должен предоставить новый URL ресурса
	304	Not modified	Используется для условных ответов, в которых тело не нужно отправлять снова.

Категория	Код	Наименование	Описание
4xx (ошибка клиента)	400	Bad request	Запрос не может быть обработан, исправьте ошибку
	401	Unauthorized	Этот код показывает, что вы не авторизированы
	403	Forbidden	Вы авторизированы, но ваши идентификационные данные не имеют прав доступа
	404	Not found	Возможно, самый узнаваемый код HTTP-статуса, так как часто встречается на странице ошибки

Категория	Код	Наименование	Описание
5xx (ошибка сервера)	500	Internal server error	Запрос не удалось выполнить из-за ошибки сервера
	501	Not implemented	Сервер не распознает запрос (чаще всего из-за метода HTTP, который неизвестен серверу)
	502	Bad gateway	Сервер, выступая в роли шлюза или прокси-сервера, получил сообщение об ошибке от вышестоящего сервера
	503	Service unavailable	По техническим причинам, например перегрузка, сервер временно не может обрабатывать запросы

Коды HTTP- ответов

Можно заметить, что некоторые коды (203, 303, 402) из более ранних версий HTTP / 1.0 RFC здесь отсутствуют.

Некоторые дополнительные коды были исключены из окончательного опубликованного RFC.

Коды HTTP- ответов

Однако несколько из них вернулось в спецификации в HTTP/1.1, но уже с другими описаниями и значениями.

Администрация адресного пространства Internet (Internet Assigned Numbers Authority, IANA) поддерживает полный список кодов состояния HTTP во всех версиях HTTP.

Коды состояния, представленные выше, впервые были определены в HTTP/1.0 1 и сейчас используются наиболее часто.

Коды HTTP- ответов

В некоторых случаях ответы могут совпадать. Например, какой код ответа вы получите в случае, если запрос не будет распознан сервером, 400 (неверный запрос) или 501 (запрос не реализован)?

Коды HTTP- ответов

Существует большое разнообразие категорий кодов ответа, так что каждое приложение может использовать наиболее подходящую из них. В спецификации сказано, что список кодов ответов можно пополнить, поэтому по мере необходимости можно добавлять новые коды и для этого не обязательно менять сам протокол.

Коды HTTP- ответов

Это еще одна причина, по которой коды ответов делятся на категории.

Новый код ответа (например, 504) может быть не распознан существующим HTTP/1.0 клиентом, однако ему будет понятно, что по какой-то причине на стороне сервера произошла ошибка, и сможет обработать его так же, как он обрабатывает другие коды ответа категории 5xx.

Заголовки HTTP- ответов

После первой строки возврата идет определенное количество (ноль или более) строк ответа заголовка HTTP/1.0.

Заголовки запросов и ответов формируются согласно одному и тому же формату.

За ними следуют два символа возврата, а затем содержимое тела (жирным шрифтом):

Заголовки HTTP- ответов

GET /

HTTP/1.0 302 Found

Location: http://www.google.ie/?gws_rd=cr&dcr=0&ei=BWe1WYrf123456qpIbwDg
Cache-

Control: private

Content-Type: text/html; charset=UTF-8

Date: Sun, 10 Sep 2017 16:23:33 GMT

Server: gws

Content-Length: 268

X-XSS-Protection: 1; mode=block

X-Frame-Options: SAMEORIGIN

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">

<TITLE>302 Moved</TITLE></HEAD><BODY>

Заголовки HTTP- ответов

Если предыдущая опубликованная версия HTTP/0.9 позволяла лишь извлекать статические документы из репозитория, то с появлением новой версии HTTP/1.0 синтаксис HTTP значительно расширился, и теперь стало возможным создание динамических, многофункциональных приложений.

Кроме того, HTTP стал сложнее, а объем спецификации протокола расширился от 700 (как было в HTTP/0.9) до 20 000 слов (в HTTP/1.0 RFC).

HTTP/1.1

Согласно системе управления версиями, HTTP/1.1 была по большей части модификацией HTTP/1.0 и не несла в себе радикальных изменений структуры протокола.

Переход от 0.9 к 1.0 включал в себя гораздо больше изменений, так как были созданы HTTP-заголовки.

HTTP/1.1

HTTP/1.1 внёс некоторые дополнительные улучшения и позволил оптимизировать использование протокола HTTP (например, постоянные соединения, обязательные заголовки сервера, улучшенные параметры кеширования и фрагментированное кодирование).

Но – что, наверное, важнее всего – на ее основе был создан официальный стандарт, на котором строилось будущее Всемирной паутины.

Обязательный заголовок HOST

URL-адрес в строках HTTP-запроса (например, команда GET), является не абсолютным URL-адресом (например, http://www.example.com/section/page.html), а относительным URL (например, /section/page.html).

Обязательный заголовок HOST

При создании HTTP предполагалось, что веб-сервер будет содержать только один вебсайт, хотя, возможно, на этом сайте будет много разделов и страниц.

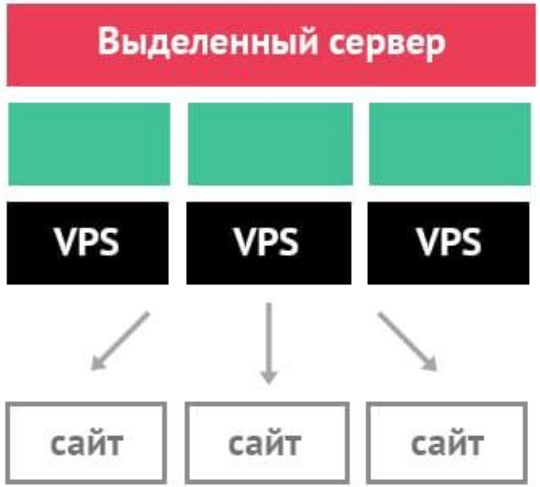
Обязательный заголовок HOST

В настоящее время на многих веб-серверах может быть расположено сразу несколько сайтов (виртуальный хостинг), поэтому важно сообщить серверу, какой именно сайт вы хотите открыть, а также какой относительный URL-адрес вам нужен.

Эту функцию можно было бы реализовать, изменив URL-адрес в HTTP-запросах на полный, абсолютный URL-адрес, но считалось, что многие существующие веб-серверы и клиенты не смогут распознать их.

Виртуальный хостинг

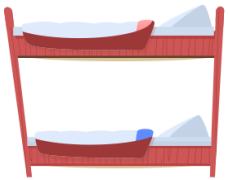
VPS



Выделенный сервер

Виртуальный хостинг (shared)

Кому подходит:
маленьким недорогим проектам, сайтам-визиткам.



Кровать в хостеле

Выделенный виртуальный сервер (VPS)

Кому подходит:
интернет-магазинам и медийным сайтам со средним трафиком.

Отдельная комната в апартаментах

Выделенный физический сервер (dedicated)

Кому подходит:
крупным ресурсам с огромным трафиком.

Дом с видом на море

Облачный хостинг

Кому подходит:
всем

Клубная карта сети отелей

Обязательный заголовок HOST

Вместо этого проблема была решена путем добавления заголовка Host:

GET / HTTP/1.1

Host: www.google.com

Обязательный заголовок HOST

В отличие от HTTP/1.0 в HTTP/1.1 его наличие стало обязательным.

Следующий запрос сформирован технически некорректно, так как он указывает на версию (HTTP/1.1), но не содержит заголовок Host:

GET / HTTP/1.1

Обязательный заголовок HOST

Тот факт, что вместо изменения относительного URL-адреса на абсолютный в спецификации прописывалось наличие обязательного поля заголовка Host, привел к появлению разногласий.

Обязательный заголовок HOST

HTTP-прокси, введенные вместе с HTTP/1.1, позволяли подключаться к HTTP-серверу через посреднический HTTP-сервер.

Синтаксис прокси-серверов требовал полных абсолютных URL-адресов для всех запросов, но на действующих веб-серверах (так называемых серверах-источниках) использовались заголовки Host.

Принцип работы PROXY

Постоянные соединения (Keep -Alive)

В HTTP/1.0 были введены постоянные соединения. Это изменение было довольно значимым и, кроме того, поддерживалось многими серверами, несмотря на то что в спецификации этой версии постоянных соединений еще не было.

Изначально HTTP представлял собой примитивный протокол типа «запрос–ответ». Клиент устанавливает соединение, запрашивает ресурс, получает ответ, и соединение закрывается.

Постоянные соединения (Keep -Alive)

Отображение одной страницы требовало использования нескольких HTTP-ресурсов, поэтому закрытие соединения (которое потом снова нужно будет открыть) вызывало ненужные задержки.

Проблема была решена путем введения нового HTTP-заголовка Connection, совместимого с версией HTTP/1.0.

Постоянные соединения (Keep -Alive)

Значение Keep-Alive в этом заголовке позволяет запросить сервер сохранить соединение открытым, чтобы разрешить отправку дополнительных запросов:

GET /page.html HTTP/1.0

Connection: Keep-Alive

Постоянные соединения (Keep -Alive)

При использовании постоянных соединений понять, когда ответ завершен, бывает довольно сложно; закрытие соединения является отчетливым признаком того, что сервер мог отправить ответ несуществующему соединению!

HTTP-заголовок Content-Length предназначен для определения длины тела ответа. Когда ответ получен полностью, клиент может приступать к отправке следующего запроса.

Постоянные соединения (Keep -Alive)

В HTTP/1.1 процесс постоянного подключения был добавлен в официальный стандарт и с тех пор выполняется по умолчанию.

Любое HTTP/1.1 соединение носит постоянный характер, даже если ответы не содержат заголовка **Connection: Keep-Alive**.

Если сервер намерен закрыть соединение, он должен включить в ответ заголовок **Connection: close**:

Постоянные соединения (Keep -Alive)

HTTP/1.1 200 OK

Date: Sun, 25 Jun 2017 13:30:24 GMT

Connection: close

Content-Type: text/html; charset=UTF-8

Server: Apache

<!doctype html>

<html>

<head>

...и т. д.

Другие новые функции HTTP /1.1

В спецификацию HTTP/1.1 было введено множество новых функций:

- к методам GET, POST и HEAD (введенных еще в HTTP/1.0) добавились новые.

Другие новые функции HTTP /1.1

Среди них PUT, OPTIONS и менее используемые CONNECT, TRACE, DELETE; методы кеширования.

Они позволяли серверу поручить клиенту сохранить ресурс (например, CSS-файл) в кеше браузера, чтобы он ог быть повторно использован позже, если потребуется.

- файлы HTTP-куки, благодаря которым HTTP перестал быть протоколом без сохранения состояния;

Другие новые функции HTTP /1.1

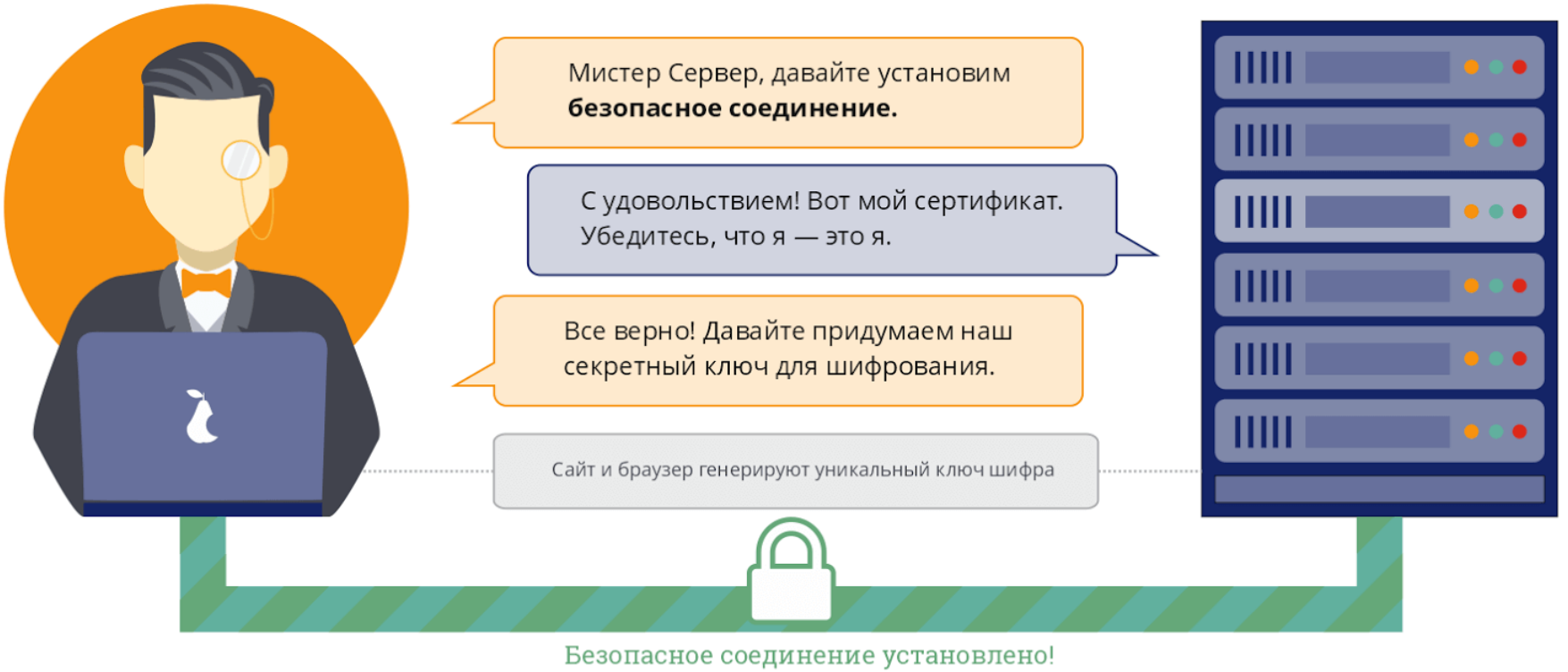
- объявление рабочей кодировки и языка HTTP-ответов;
- поддержка прокси;
- аутентификация;
- новые коды состояния;
- завершающие заголовки

HTTPS

Поскольку HTTP-сообщение – это обычный текст, злоумышленникам не составит труда перехватить его, прочитать или даже изменить содержание.

HTTPS является защищенной версией HTTP, которая шифрует передаваемые сообщения с помощью протокола Transport Layer Security (TLS).

Предыдущая версия носила название Secure Sockets Layer (SSL),


Мистер Сервер, давайте установим **безопасное соединение.**
С удовольствием! Вот мой сертификат. Убедитесь, что я — это я.
Все верно! Давайте придумаем наш секретный ключ для шифрования.
Сайт и браузер генерируют уникальный ключ шифра
Безопасное соединение установлено!

HTTPS

HTTPS добавляет к HTTP три важных аспекта:

- шифрование (третьи лица не смогут прочитать сообщения во время их передачи);
- целостность (сообщение невозможно изменить в процессе передачи, так как зашифрованное сообщение имеет цифровую подпись, и эта подпись криптографически проверяется перед расшифровкой);
- аутентификация (обеспечивает передачу сообщения именно на нужный сервер).

SSL, TLS, HTTPS и HTTP

HTTPS зашифровывает сообщения по протоколам SSL и TLS.

SSL был разработан компанией Netscape. Версия протокола SSLv1 так и не увидела свет, и вместо нее в 1995 году Netscape выпустила версию SSLv2.

В 1996 году компания выпустила SSLv3, в которой были исправлены некоторые недостатки предыдущих версий.

SSL, TLS, HTTPS и HTTP

SSL не считался официальным Internet-стандартом, поскольку принадлежал компании Netscape, но впоследствии IETF опубликовала его постфактум как официальный документ.

Далее на основании SSL был оформлен и стандартизирован новый протокол, получивший название TLS.

SSL, TLS, HTTPS и HTTP

Протоколы TLSv1.0 и SSLv3 были весьма похожи, однако все же несовместимы.

Версии TLSv1.1 и TLSv1.2 d были созданы в 2006 и 2008 годах соответственно.

Они стали более безопасными.

TLSv1.3 был признан стандартом в 2018 году

SSL, TLS, HTTPS и HTTP

Несмотря на то что были созданы новые и более безопасные стандартизированные версии, многие люди все еще пользовались SSLv3, поэтому он долгое время оставался стандартом де-факто, хотя многие клиенты поддерживали и TLSv1.0.

SSL, TLS, HTTPS и HTTP

Однако в 2014 году в SSLv3 были обнаружены серьезные уязвимости, в связи с чем он перестал поддерживаться браузерами и вышел из употребления.

Эта ситуация положила начало развитию TLS. Спустя некоторое время в TLSv1.0 обнаружили аналогичные уязвимости.

Совет по стандартам безопасности призывал использоватьTLSv1.1 или более поздние версии.

SSL, TLS, HTTPS и HTTP

В итоге люди стали путать названия протоколов. Многие до сих пор говорят «SSL», потому что он оставался стандартом очень долгое время, а другие говорят «SSL/TLS» или «TLS».

Чтобы избежать недоразумений, иногда говорят просто «HTTPS», хотя этот термин и не совсем корректен.

SSL, TLS, HTTPS и HTTP

SSL, TLS, HTTPS и HTTP

HTTPS работает с использованием шифрования с открытым ключом, которое позволяет серверам предоставлять пользователям открытые ключи в виде цифровых сертификатов при первом подключении.

SSL, TLS, HTTPS и HTTP

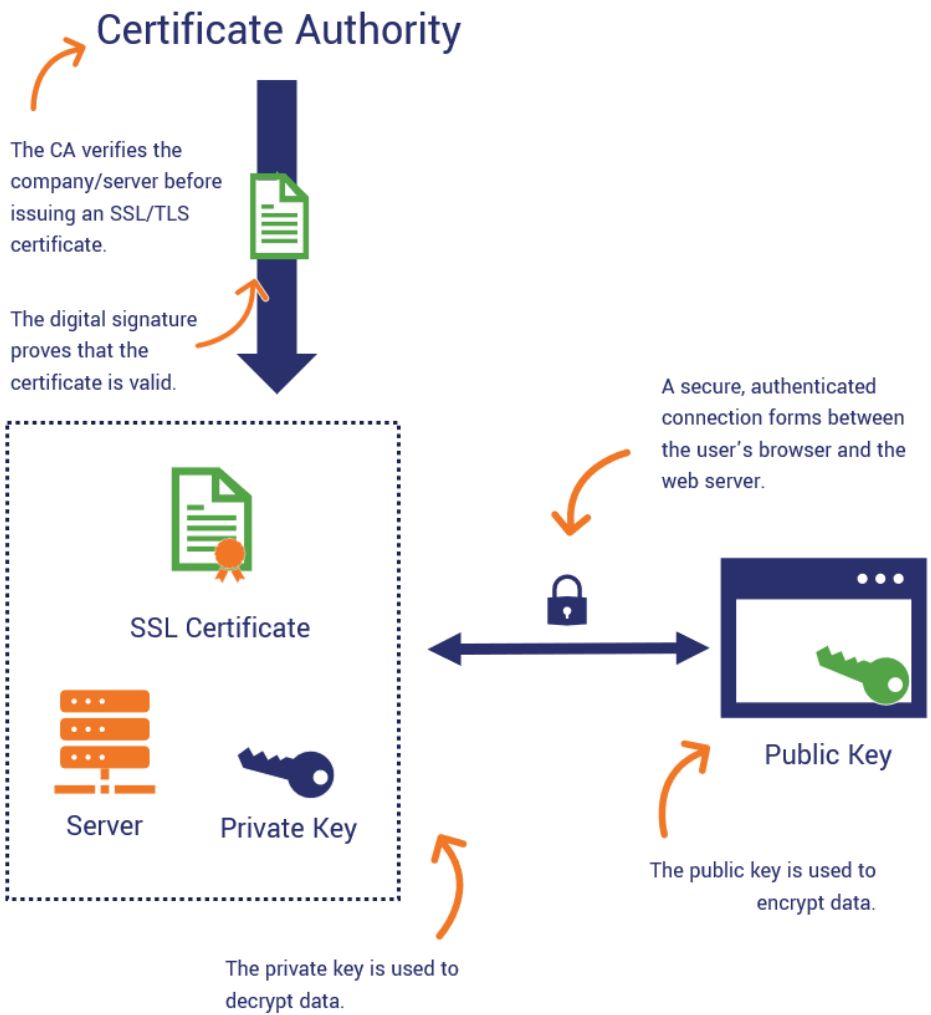
Браузер шифрует сообщения с помощью этого открытого ключа, расшифровать который может только сервер-получатель, так как только он имеет соответствующий секретный ключ.

Такая система обеспечивает вам безопасное взаимодействие с веб-сайтом без необходимости заранее знать общий секретный ключ.

SSL, TLS, HTTPS и HTTP

Цифровые сертификаты выдаются различными центрами сертификации (Certification authority), с которыми работают веб-браузеры, а также подписываются ими цифровой подписью, поэтому проверить подлинность открытого ключа для сервера, к которому вы подключаетесь, очень легко.

Проблема заключается в том, что HTTPS показывает, что вы подключаетесь к серверу, но не гарантирует его безопасности.


Certificate Authority
The CA verifies the company/server before issuing an SSL/TLS certificate.
The digital signature proves that the certificate is valid.
SSL Certificate
Server
Private Key
A secure, authenticated connection forms between the user's browser and the web server.
Public Key
The public key is used to encrypt data.
The private key is used to decrypt data.


HTTP
HTTPS DV and OV certificates
HTTPS EV certificates
Chrome
Not secure
bbc.com
tunetheweb.com
Twitter, Inc. [US]
twitter.com
Firefox
www.bbc.com
https://www.tunetheweb.com
Twitter, Inc. (US)
https://twitter.com
Opera
www.bbc.com
www.tunetheweb.com
Twitter, Inc. [US]
twitter.com
IE 11
http://www.bbc.com/
https://www.tunetheweb.com/
https://twitter.com/
Twitter, Inc. [US]
EDGE
bbc.com
tunetheweb.com
Twitter, Inc. [US]
twitter.com

Сертификаты браузеров

Status	Chrome	Firefox	Edge	Safari	Opera	Notes
HTTP						No indicator shown.
Mixed HTTPS (active content)						Some browsers warn against mixed active content.
Mixed HTTPS (passive content)						All browsers show a positive indicator. Most commonly a lock icon.
HTTPS (DV)						All browsers show a positive indicator.
HTTPS (OV)						All browsers show a positive indicator.
HTTPS (EV)						All browsers show a positive indicator, but most browsers show the verified name of the organization in a green color.

SSL, TLS, HTTPS и HTTP

Протокол HTTPS разработан на основе HTTP и очень схож с ним. По умолчанию у него другой порт (порт 443, в отличие от стандартного порта 80 для HTTP), а также он имеет другую схему URL-адресов (https://, а не http://).

Однако эти аспекты не столь значительны, так как HTTPS и HTTP имеют одинаковый синтаксис и формат сообщений, а отличаются они лишь в планах шифрования и дешифрования.

SSL, TLS, HTTPS и HTTP

Когда клиент подключается к HTTPS-серверу, он проходит через стадию согласования (или TLS-рукопожатия).

Во время этого процесса сервер предоставляет открытый ключ, клиент и сервер согласовывают используемые методы шифрования, а затем клиент и сервер согласовывают общий ключ шифрования, чтобы использовать его в будущем.

SSL, TLS, HTTPS и HTTP

Криптография с открытым ключом работает медленно, поэтому открытые ключи шифрования применяются только для согласования подключения сервера и клиента, который используется для шифрования будущих сообщений, что обеспечивает лучшую производительность.

SSL, TLS, HTTPS и HTTP

После создания сессии HTTPS происходит обмен стандартными HTTP-сообщениями. Клиент и сервер зашифровывают их перед отправкой и расшифровывают при получении, однако для обычного веб-разработчика или менеджера сервера между HTTPS и HTTP нет уже фактически никакой разницы.

SSL, TLS, HTTPS и HTTP

Все процессы происходят прозрачно, если вы, конечно, не смотрите на необработанные сообщения, отправленные по сети.

При переходе на HTTPS мы продолжаем пользоваться стандартными HTTP-запросами и ответами, а не заменяем их каким-то другим протоколом.

SSL, TLS, HTTPS и HTTP

Веб-серверам, использующим HTTPS, необходим клиент с поддержкой HTTPS, выполняющий шифрование и дешифрование. Telnet уже не подходит для этой задачи, поэтому для отправки примеров HTTP-запросов на эти серверы следует выбрать другой клиент.

SSL, TLS, HTTPS и HTTP

Возможно использовать команду s_client в OpenSSL, которая позволит вам отправлять HTTP-команды на HTTPS-сервер аналогично тому, это делается с помощью Telnet:

openssl s_client -crlf -connect www.google.com:443 -quiet

GET / HTTP/1.1

Host: www.google.com

HTTP/1.1 200 OK

Инструменты для просмотра, отправки и получения HTTP-сообщений

Сегодня все веб-браузеры оснащены так называемыми инструментами разработчика, с помощью которых вы можете увидеть некоторые детали устройства и работы веб-сайтов, включая HTTP-запросы и ответы.

Инструменты для просмотра, отправки и получения HTTP-сообщений

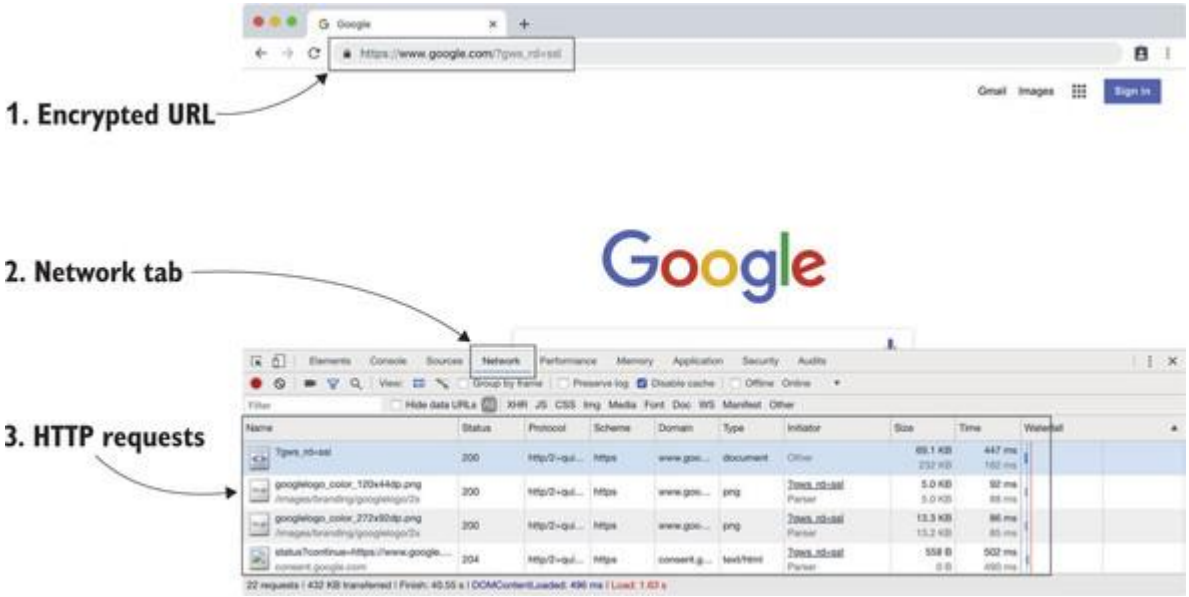
Инструменты разработчика запускаются сочетанием клавиш (F12 в большинстве браузеров для ОС Windows или Option+Command+I на компьютерах Apple).

Так же вы можете кликнуть на элемент страницы правой кнопкой мыши и во всплывающем контекстном меню выбрать

Инструменты для просмотра, отправки и получения HTTP-сообщений

Посмотреть код. Панель инструментов разработчика содержит несколько вкладок, в которых вы можете увидеть различные технические детали строения веб-страниц, однако сейчас нас больше всего интересует вкладка Network (Сеть).

Инструменты для просмотра, отправки и получения HTTP-сообщений



Инструменты для просмотра, отправки и получения HTTP-сообщений

URL-адрес вводится в верхней части адресной строки (1). Обратите внимание на значок замочка и https://. Это означает, что Google использует HTTPS (хотя, как уже упоминалось, Chrome может перестать использовать его).

Инструменты для просмотра, отправки и получения HTTP-сообщений

Веб-страница размещается под адресной строкой. Однако если вы откроете панель инструментов разработчика, то увидите на странице новый раздел, содержащий различные вкладки.

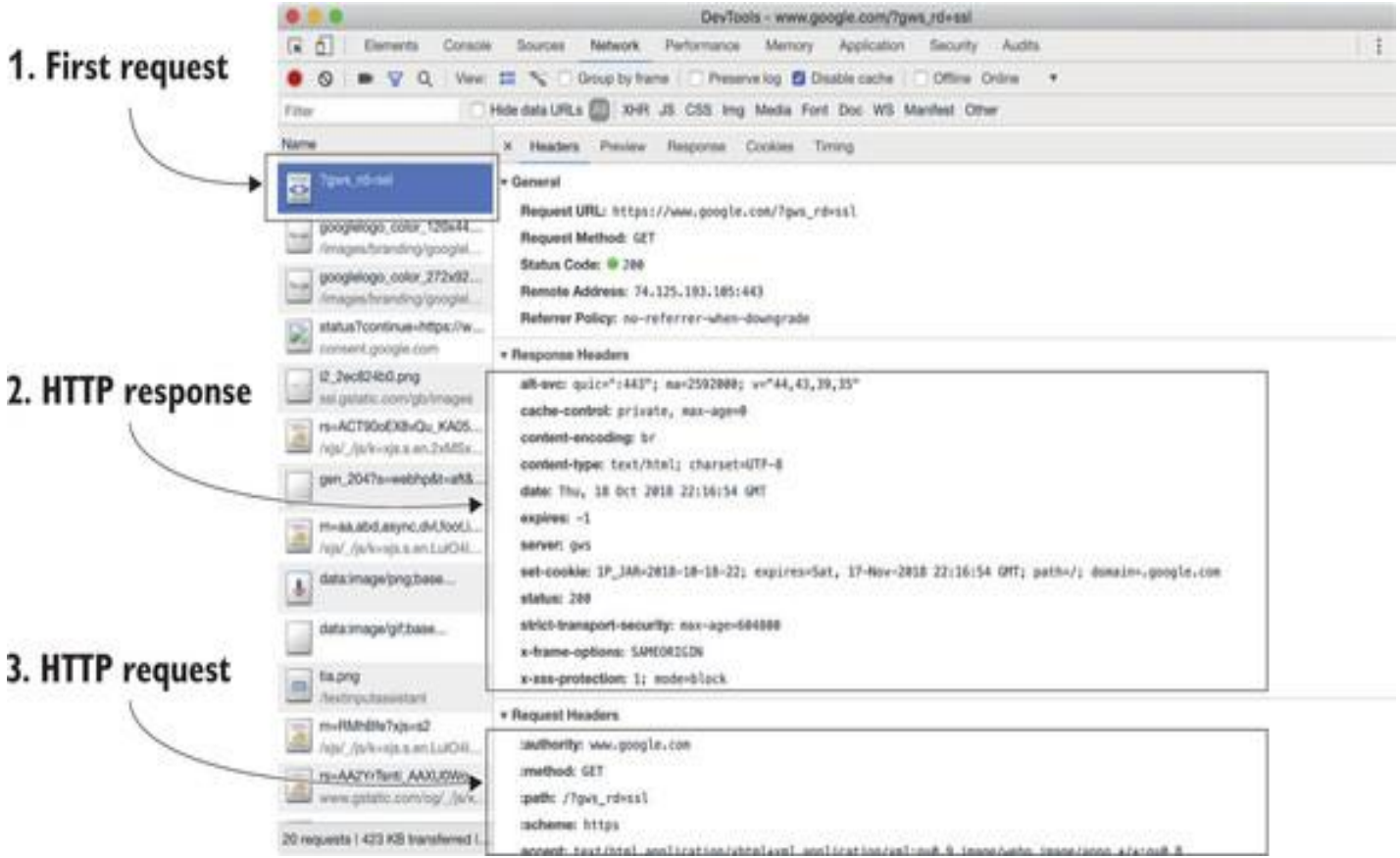
Инструменты для просмотра, отправки и получения HTTP-сообщений

При нажатии на вкладку Сеть (2) отображаются HTTP-запросы (3), включая такую информацию, как HTTP метод (GET), статус ответа (200), версия протокола (http/1.1) и схема (https).

Инструменты для просмотра, отправки и получения HTTP-сообщений

Вы можете изменить отображаемые столбцы, кликнув правой кнопкой мыши на их заголовки. Например, столбцы Protocol (Протокол), Scheme (Схема) и Domain (Домен) не отображаются по умолчанию, и на некоторых веб-сайтах (например, Twitter) в столбце для HTTP/2 вы можете увидеть h2, а для более новой версии протокола, возможно, даже http/2+quic (Google).

Что происходит при нажатии на первый запрос?



Инструменты для просмотра, отправки и получения HTTP-сообщений

Браузер обрабатывает HTTPS-запросы, поэтому инструменты разработчика показывают сообщения запросов до их шифрования и ответные сообщения после их дешифрования.

Инструменты для просмотра, отправки и получения HTTP-сообщений

Как правило, если у вас есть правильные инструменты для обработки шифрования и дешифрования сообщений, после настройки HTTPS-соединения дальнейшая работа протокола уже мало чем интересна.

Инструменты для просмотра, отправки и получения HTTP-сообщений

Кроме того, инструменты разработчика в большинстве веб-браузеров хорошо справляются с задачей отображения медиафайлов, а код (HTML, CSS, и JavaScript) можно отформатировать, чтобы его было легче читать.

Отправка HTTP-запросов

Использование инструментов разработчика веб-браузеров является наилучшим способом просмотра еще не обработанных HTTP-запросов и ответов, однако – что весьма удивительно – отправлять такие запросы эти инструменты не могут.

Отправка HTTP-запросов

Инструменты разработчика в браузере редко позволяют отправлять необработанные HTTP-сообщения, но, например, с помощью адресной строки можно отправлять простые GET-запросы, а некоторые веб-сайты предоставляют определенные функциональные возможности, например отправку POST-запросов с помощью HTML-форм.

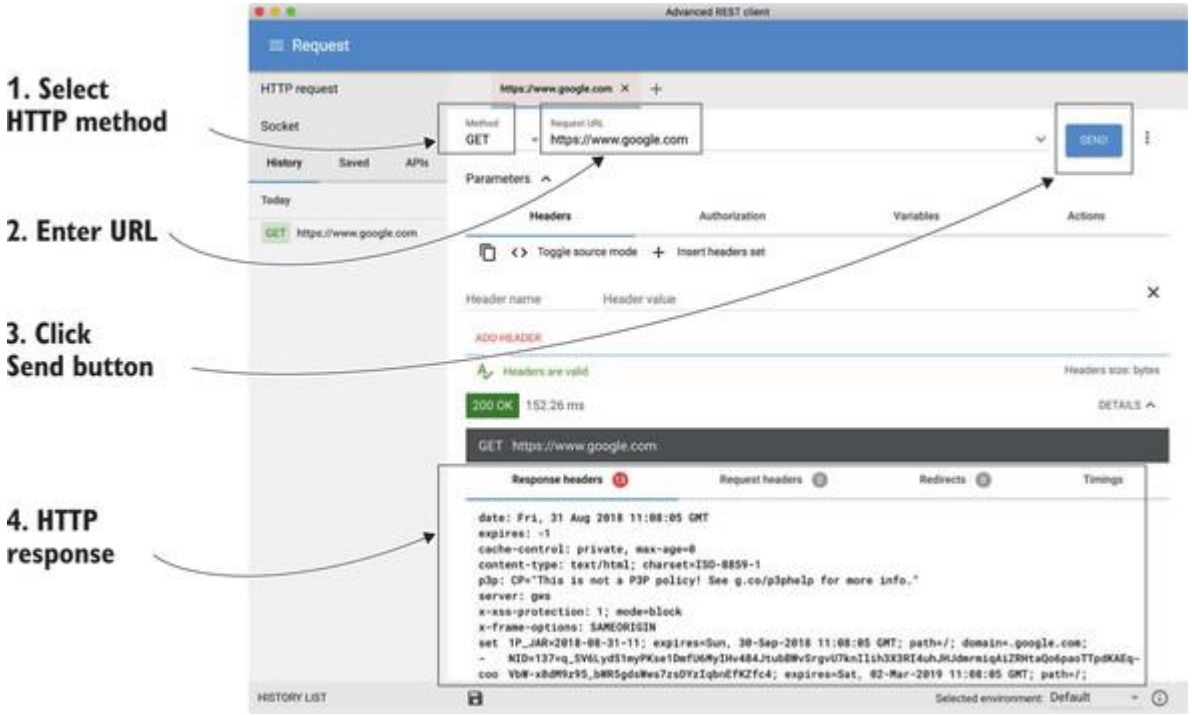
Отправка HTTP-запросов

Расширение для браузеров Advanced REST Client дает вам возможность отправлять необработанные HTTP-сообщения и просматривать ответы.

Отправьте запрос GET (1) для URL-адреса https://www.google.com (2) и нажмите Отправить (3), затем вы получите ответ (4), как показано на рисунке.

Обратите внимание, что это приложение также обрабатывает запросы и ответы с помощью HTTPS.

Что происходит при нажатии на первый запрос?



Отправка HTTP-запросов

Работа с расширением Advanced Rest Client очень проста и привычна, однако при этом оно позволяет отправлять многие типы HTTP-запросов (например, POST и PUT), а также создавать заголовок или тело данных, предназначенных для отправки.

Отправка HTTP-запросов

Изначально Advanced REST Client был расширением для Chrome, но позже его оформили как отдельное приложение. Существуют и другие подобные расширения, имеющие соответствующую функциональность, например Postman (Chrome), Rested, RESTClient (Firefox) и RESTMan (Opera).

Другие инструменты для просмотра и отправки HTTP-запросов

Вне браузера вам также доступно множество других инструментов для отправки или просмотра HTTP-запросов. Среди них и инструменты командной строки (например, curl, wget и httpie), и настольные клиенты например, SOAP-UI). Если вы хотите ознакомиться с сетевой информацией, зайдите на страницу net-internals в Chrome или воспользуйтесь анализаторами трафика, такими как Fiddler и Wireshark.