HW

Client\_Server

1) Прочиать про клиент-серверную архитектуру

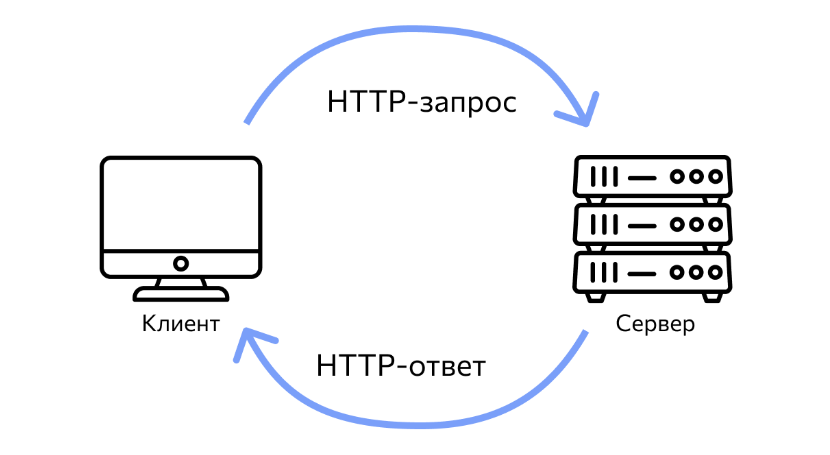
client–server) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, передача файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

2) Что ткое HTTP и HTTPS

HTTP представляет собой прикладной протокол 7 уровня. Действие осуществляется в следующем порядке: клиент направляет запрос к серверу расположения ресурса, протокол форматирует данные нужным образом и предоставляет результат, затем браузер отображает полученные данные. В общем, HTTP – это набор правил передачи информации между браузером пользователя и сервером сайта. В основе его работы находится существующая клиент-серверная передача данных:

Пользователь составляет и вводит запрос

Серверное программное обеспечение обрабатывает запрос, систематизирует его, преобразует, а далее отправляет пользователю.



HTTP-запрос состоит из трех элементов:

стартовой строки, которая задает параметры запроса или ответа,

заголовка, который описывает сведения о передаче и другую служебную информацию.

тело (его не всегда можно встретить в структуре). Обычно в нем как раз лежат передаваемые данные. От заголовка тело отделяется пустой строкой.

Важнейшим элементом структуры запроса является стартовая строка. Благодаря ей сервер понимает, что от него хотят. Вот как она устроена:

Метод + URL + HTTP/Версия

HTTPS поддерживает шифрование с повышенным уровнем безопасного обмена данными. Домены, на которых установлен SSL-сертификат, являются платформой для применения данного расширения. Технологии для шифрования применяются во всех этапах обмена информации. Преимущества следующие:

1. Гарантированная безопасность — это решающий фактор при работе с конфиденциальными и финансовыми инструментами. Все ресурсы, работающие с обработкой персональных и финансовых данных работают по протоколу HTTPS;

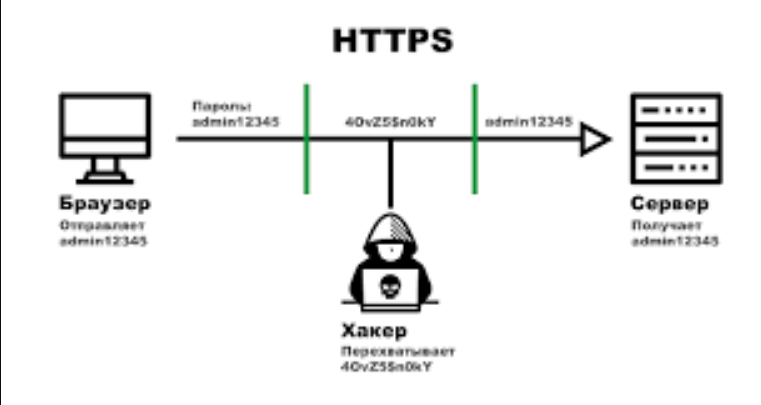
2. Защита от хакерских атак, направленных на прослушивание соединения;

3. Поисковые системы настоятельно рекомендуют всем ресурсам перейти на HTTPS протокол;

4. Скорость загрузки данных практически одинакова с показателем HTTP. Безопасность важнее, чем несущественный прирост скорости передачи данных.

По протоколу HTTPS в большинстве случаев работают: интернет-магазины, сервисы-кэшбэк, платежные ресурсы с электронными кошельками, различные финансовые подразделения, транспортные предприятия, государственные сайты и другие подобные ресурсы.

На данный момент, фактически, использование HTTPS с сертификатом SSL — стандарт «де-факто» практически для любого сайта.



3) HTTP методы

Метод (иногда его называют HTTP-глаголом) – описывает, какое именно действие нужно совершить со страницей. Можно придумать самые разные, но стандартных методов девять: GET, HEAD, POST, PUT, DELETE,CONNECT, OPTIONS, TRACE, PATCH. Их функциональность раскрывается в названии, они позволяют получить данные (GET), отправить данные на сервер (POST), удалить (DELETE) или заменить часть (PATCH).

Чаще всего используют GET и POST, они нужны для чтения и отправки данных на сервер. Например вы зашли в соцсеть, увидели пост и решили оставить комментарий. Или зашли в интернет-магазин, решили что-то купить и оставили данные карты.



4) HTTP статус коды сервера

Информационные 100 - 199

Успешные 200 - 299

Перенаправления 300 - 399

Клиентские ошибки 400 - 499

Серверные ошибки 500 - 599

| Код ответа | Название | Описание | Версия HTTP |
| --- | --- | --- | --- |
| Информационные | | | |
| 100 | Continue | "Продолжить". Этот промежуточный ответ указывает, что запрос успешно принят и клиент может продолжать присылать запросы либо проигнорировать этот ответ, если запрос был завершён. | Только HTTP/1.1 |
| 101 | Switching Protocol | "Переключение протокола". Этот код присылается в ответ на запрос клиента, содержащий заголовок Upgrade:, и указывает, что сервер переключился на протокол, который был указан в заголовке. Эта возможность позволяет перейти на несовместимую версию протокола и обычно не используется. | Только HTTP/1.1 |
| 102 | Processing | "В обработке". Этот код указывает, что сервер получил запрос и обрабатывает его, но обработка ещё не завершена. | Только HTTP/1.1 |
| 103 | Early Hints | "Ранние подсказки". В ответе сообщаются ресурсы, которые могут быть загружены заранее, пока сервер будет подготавливать основной ответ. [RFC 8297 (Experimental)](https://datatracker.ietf.org/doc/rfc8297). | Только HTTP/1.1 |
| Успешные | | | |
| 200 | OK | "Успешно". Запрос успешно обработан. Что значит "успешно", зависит от метода HTTP, который был запрошен:  GET: "ПОЛУЧИТЬ". Запрошенный ресурс был найден и передан в теле ответа.  HEAD: "ЗАГОЛОВОК". Заголовки переданы в ответе.  POST: "ПОСЫЛКА". Ресурс, описывающий результат действия сервера на запрос, передан в теле ответа.  TRACE: "ОТСЛЕЖИВАТЬ". Тело ответа содержит тело запроса полученного сервером. | HTTP/0.9 и выше |
| 201 | Created | "Создано". Запрос успешно выполнен и в результате был создан ресурс. Этот код обычно присылается в ответ на запрос PUT "ПОМЕСТИТЬ". | HTTP/0.9 и выше |
| 202 | Accepted | "Принято". Запрос принят, но ещё не обработан. Не поддерживаемо, т.е., нет способа с помощью HTTP отправить асинхронный ответ позже, который будет показывать итог обработки запроса. Это предназначено для случаев, когда запрос обрабатывается другим процессом или сервером, либо для пакетной обработки. | HTTP/0.9 и выше |
| 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. Во всех остальных ситуациях более предпочтителен код ответа 200 OK. | HTTP/0.9 и 1.1 |
| 204 | No Content | "Нет содержимого". 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. | HTTP/0.9 и выше |
| 205 | Reset Content | "Сбросить содержимое". Этот код присылается, когда запрос обработан, чтобы сообщить клиенту, что необходимо сбросить отображение документа, который прислал этот запрос. | Только HTTP/1.1 |
| 206 | Partial Content | "Частичное содержимое". Этот код ответа используется, когда клиент присылает заголовок диапазона, чтобы выполнить загрузку отдельно, в несколько потоков. | Только HTTP/1.1 |
| Сообщения о перенаправлениях | | | |
| 300 | Multiple Choice | "Множественный выбор". 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. | HTTP/1.0 и выше |
| 301 | Moved Permanently | "Перемещён на постоянной основе". Этот код ответа значит, что URI запрашиваемого ресурса был изменён. Возможно, новый URI будет предоставлен в ответе. | HTTP/0.9 и выше |
| 302 | Found | "Найдено". 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. | HTTP/0.9 и выше |
| 303 | See Other | "Просмотр других ресурсов". Этот код ответа присылается, чтобы направлять клиента для получения запрашиваемого ресурса в другой URI с запросом GET. | HTTP/0.9 и 1.1 |
| 304 | Not Modified | "Не модифицировано". Используется для кеширования. Это код ответа значит, что запрошенный ресурс не был изменён. Таким образом, клиент может продолжать использовать кешированную версию ответа. | HTTP/0.9 и выше |
| 305 | Use Proxy | "Использовать прокси". Это означает, что запрошенный ресурс должен быть доступен через прокси. Этот код ответа в основном не поддерживается из соображений безопасности. | Только HTTP/1.1 |
| 306 | Switch Proxy | Больше не использовать. Изначально подразумевалось, что " последующие запросы должны использовать указанный прокси." | Только HTTP/1.1 |
| 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: если в первом запросе использовался POST, то во втором запросе также должен использоваться POST. | Только HTTP/1.1 |
| 308 | Permanent Redirect | "Перенаправление на постоянной основе". Это означает, что ресурс теперь постоянно находится в другом URI, указанном в заголовке Location: HTTP Response. Данный код ответа имеет ту же семантику, что и код ответа 301 Moved Permanently, за исключением того, что агент пользователя не должен изменять используемый метод HTTP: если POST использовался в первом запросе, POST должен использоваться и во втором запросе.  Примечание: Это экспериментальный код ответа, Спецификация которого в настоящее время находится в черновом виде. | [draft-reschke-http-status-308](http://greenbytes.de/tech/webdav/draft-reschke-http-status-308-07.html) |
| Клиентские | | | |
| 400 | Bad Request | "Плохой запрос". Этот ответ означает, что сервер не понимает запрос из-за неверного синтаксиса. | HTTP/0.9 и выше |
| 401 | Unauthorized | "Неавторизованно". Для получения запрашиваемого ответа нужна аутентификация. Статус похож на статус 403, но,в этом случае, аутентификация возможна. | HTTP/0.9 и выше |
| 402 | Payment Required | "Необходима оплата". Этот код ответа зарезервирован для будущего использования. Первоначальная цель для создания этого когда была в использовании его для цифровых платёжных систем(на данный момент не используется). | HTTP/0.9 и 1.1 |
| 403 | Forbidden | "Запрещено". У клиента нет прав доступа к содержимому, поэтому сервер отказывается дать надлежащий ответ. | HTTP/0.9 и выше |
| 404 | Not Found | "Не найден". Сервер не может найти запрашиваемый ресурс. Код этого ответа, наверно, самый известный из-за частоты его появления в вебе. | HTTP/0.9 и выше |
| 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. | Только HTTP/1.1 |
| 406 | Not Acceptable | Этот ответ отсылается, когда веб сервер после выполнения [server-driven content negotiation](https://developer.mozilla.org/en-US/HTTP/Content_negotiation" \l "server-driven_negotiation" \o "https://developer.mozilla.org/en/HTTP/Content_negotiation#Server-driven_negotiation), не нашёл контента, отвечающего критериям, полученным из user agent. | Только HTTP/1.1 |
| 407 | Proxy Authentication Required | Этот код ответа аналогичен коду 401, только аутентификация требуется для прокси сервера. | Только HTTP/1.1 |
| 408 | Request Timeout | Ответ с таким кодом может прийти, даже без предшествующего запроса. Он означает, что сервер хотел бы отключить это неиспользуемое соединение. Этот метод используется все чаще с тех пор, как некоторые браузеры, вроде Chrome и IE9, стали использовать [HTTP механизмы предварительного соединения](http://www.belshe.com/2011/02/10/the-era-of-browser-preconnect/) для ускорения сёрфинга  (смотрите [баг 634278](https://bugzilla.mozilla.org/show_bug.cgi?id=634278), будущей реализации этого механизма в Firefox). Также учитывайте, что некоторые серверы прерывают соединения не отправляя подобных сообщений. | Только HTTP/1.1 |
| 409 | Conflict | Этот ответ отсылается, когда запрос конфликтует с текущим состоянием сервера. | Только HTTP/1.1 |
| 410 | Gone | Этот ответ отсылается, когда запрашиваемый контент удалён с сервера. | Только HTTP/1.1 |
| 411 | Length Required | Запрос отклонён, потому что сервер требует указание заголовка Content-Length, но он не указан. | Только HTTP/1.1 |
| 412 | Precondition Failed | Клиент указал в своих заголовках условия, которые сервер не может выполнить | Только HTTP/1.1 |
| 413 | Request Entity Too Large | Размер запроса превышает лимит, объявленный сервером. Сервер может закрыть соединение, вернув заголовок Retry-After | Только HTTP/1.1 |
| 414 | Request-URI Too Long | URI запрашиваемый клиентом слишком длинный для того, чтобы сервер смог его обработать | Только HTTP/1.1 |
| 415 | Unsupported Media Type | Медиа формат запрашиваемых данных не поддерживается сервером, поэтому запрос отклонён | Только HTTP/1.1 |
| 416 | Requested Range Not Satisfiable | Диапазон указанный заголовком запроса Range не может быть выполнен; возможно, он выходит за пределы переданного URI | Только HTTP/1.1 |
| 417 | Expectation Failed | Этот код ответа означает, что ожидание, полученное из заголовка запроса Expect, не может быть выполнено сервером. | Только HTTP/1.1 |
| Серверные | | | |
| 500 | Internal Server Error | "Внутренняя ошибка сервера". Сервер столкнулся с ситуацией, которую он не знает как обработать. | HTTP/0.9 и выше |
| 501 | Not Implemented | "Не выполнено". 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. | HTTP/0.9 и выше |
| 502 | Bad Gateway | "Плохой шлюз". Эта ошибка означает что сервер, во время работы в качестве шлюза для получения ответа, нужного для обработки запроса, получил недействительный (недопустимый) ответ. | HTTP/0.9 и выше |
| 503 | Service Unavailable | "Сервис недоступен". 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.  Этот ответ должен использоваться для временных условий и Retry-After: HTTP-заголовок должен, если возможно, содержать  предполагаемое время до восстановления сервиса. Веб-мастер также должен позаботиться о заголовках, связанных с кешем, которые отправляются вместе с этим ответом, так как эти ответы, связанные с временными условиями, обычно не должны кешироваться. | HTTP/0.9 и выше |
| 504 | Gateway Timeout | Этот ответ об ошибке предоставляется, когда сервер действует как шлюз и не может получить ответ вовремя. | Только HTTP/1.1 |
| 505 | HTTP Version Not Supported | "HTTP-версия не поддерживается". HTTP-версия, используемая в запросе, не поддерживается сервером. | Только HTTP/1.1 |

5) Что такое ядро браузера

Браузер = оболочка + ядро  
Среди них относительно больше типов оболочек и меньше ядер. Оболочка относится к оболочке браузера: например, меню, панели инструментов и т. Д. Он в основном предназначен для работы с пользовательским интерфейсом, настройки параметров и так далее. Он вызывает ядро ​​для реализации различных функций. Ядро - это ядро ​​браузера. Ядро - это программа или модуль, который отображает контент на основе языка разметки.

Что такое ядро ​​браузера

Ядро браузера можно разделить на две части: движок рендеринга (инженер макета или движок рендеринга) и движок JS. Он отвечает за получение содержимого веб-страницы (HTML, XML, изображения и т. Д.), Организацию информации (например, добавление CSS и т. Д.) И расчет режима отображения веб-страницы, а затем вывод ее на монитор или принтер. Разница в ядре браузера будет по-разному интерпретировать синтаксис веб-страницы, поэтому эффект рендеринга будет другим. Все веб-браузеры, почтовые клиенты и другие приложения, которым необходимо редактировать и отображать сетевой контент, требуют ядра. (См. Википедия). Движок JS анализирует язык Javascript и выполняет язык Javascript для достижения динамических эффектов веб-страницы. Сначала не было четкого различия между движком рендеринга и движком JS, а позже движок JS становился все более независимым, и ядро ​​имело тенденцию ссылаться только на движок рендеринга. Механизм рендеринга определяет, как браузер отображает содержимое веб-страницы и информацию о формате страницы. Разные ядра браузеров по-разному интерпретируют синтаксис записи веб-страниц, поэтому эффект рендеринга (отображения) одной и той же веб-страницы в браузерах разных ядер также может быть различным. Именно поэтому авторам веб-страниц необходимо тестировать веб-страницы в браузерах разных ядер. Покажите причину эффекта. Движок JS отвечает за интерпретацию, компиляцию и выполнение JavaScript, чтобы заставить веб-страницу достигать некоторых динамических эффектов. Но обычные ядра браузера можно разделить на эти пять типов: Trident, Gecko, Presto, Webkit, Blink.

Классификация ядра

Trident

Trident (ядро IE): Эта программа ядра была впервые принята в IE4 в 1997 году. Она была изменена Microsoft на основе кода Mosaic и продолжала использовать IE11. Она также обычно называется «ядром IE». Trident на самом деле является открытым ядром, и его интерфейсное ядро ​​достаточно развито, поэтому есть много браузеров (оболочек), которые используют IE вместо ядра IE к  
Из-за «монополии» самого IE (хотя IE не является монополией по названию, но на самом деле, особенно с Windows 95 до первых дней XP, IE действительно использовал Dongfeng Windows находится в «монопольном» положении и долгое время доминировало над ядром Trident. Microsoft долгое время не обновляла ядро ​​Trident. Это привело к двум последствиям, одно из которых состоит в том, что ядро ​​Trident почти не касалось стандарта W3C ( 2005 г.) Во-вторых, большое количество ошибок и других проблем безопасности в ядре Trident не было решено вовремя, и затем некоторые разработчики и некоторые ученые, которые привержены открытому исходному коду, открывают свое мнение, что Internet Explorer небезопасен, и многие пользователи обратились к нему. В это время появились другие браузеры, Firefox и Opera. Значительное увеличение доли рынка браузеров с ядром без Trident также заставило многих веб-разработчиков обратить внимание на веб-стандарты и эффекты просмотра браузеров без IE. к  
Дополнение: начиная с версии 11 IE изначально поддерживает технологию WebGL. Механизм JavaScript IE8 - Jscript, и IE9 начал использовать Chakra. Две версии очень разные. Chakra превосходна с точки зрения скорости и стандартизации. к  
Общие браузеры с ядром Trident: [1] IE6, IE7, IE8 (Trident 4.0), IE9 (Trident 5.0), IE10 (Trident 6.0); [1] 360 браузер безопасности (1.0-5.0 - Trident, 6.0 - Trident + Webkit, 7.0 - Trident + Blink)[Гепард Ультра Лайт Браузер](https://www.baidu.com/s?wd=%E7%8C%8E%E8%B1%B9%E6%9E%81%E8%BD%BB%E6%B5%8F%E8%A7%88%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)360-скоростной браузер (Trident + Webkit до 7.5, Trident + Blink для 7.5)[Cheetah Safe Browser](https://www.baidu.com/s?wd=%E7%8C%8E%E8%B1%B9%E5%AE%89%E5%85%A8%E6%B5%8F%E8%A7%88%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)(Версия 1.0-4.2 - Trident + Webkit, 4.3 и более поздние версии - Trident + Blink) Сверхлегкий браузер Cheetah, браузер Maxthon (Maxthon 1.x и 2.x - ядро ​​IE, 3.x - двухъядерный IE и Webkit) , Браузер Baidu (ранняя версия), браузер Window of the World [2] (изначально ядро ​​IE, ядро ​​Chrome + IE, выпущенное в 2013 году), браузер 2345, Tencent TT, браузер Taobao, браузер сбора и чтения, Sogou Express Браузер (Trident для 1.x, Trident + Webkit для 2.0 и более поздних версий), браузер Ayun (ранняя версия),[Rising Security Browser](https://www.baidu.com/s?wd=%E7%91%9E%E6%98%9F%E5%AE%89%E5%85%A8%E6%B5%8F%E8%A7%88%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank), Тонкий браузер, GreenBrowser, браузер Aifan (версия до 12), браузер 115, браузер 155, флэш-браузер, браузер оксида углерода, браузер конфет, браузер Rainbow, браузер Ruiying, Храбрый без границ Браузер, браузер 114, браузер Ant, браузер Feiteng, браузер Suda, браузер Zorro, браузер Dolphin (iPhone / iPad / Android), браузер UC (ядро Blink + ядро ​​Trident) и т. Д. к  
Новая версия некоторых браузеров является «двухъядерной» или даже «многоядерной», один из которых - Trident, а затем добавляется еще один. Отечественные производители обычно называют другие ядра «высокоскоростным режимом просмотра», тогда как Trident является «совместимым режимом просмотра», и пользователи могут переключаться между ними.

Gecko

Gecko (ядро Firefox): Netscape 6 начал внедрять ядро, а позже Mozilla FireFox (браузер Firefox) также принял это ядро. Отличительной чертой Gecko является то, что код полностью открыт, поэтому его можно разрабатывать в высокой степени. Программисты по всему миру Вы можете написать код для него и добавить функциональность. Поскольку это ядро ​​с открытым исходным кодом, многие его одобряют, и для ядра Gecko существует множество браузеров, что является важной причиной того, что ядро ​​Gecko является молодым, но его доля на рынке может быстро возрасти. к  
На самом деле происхождение движка Gecko не связано с IE. Ранее я говорил, что IE не использует стандарты W3C, что вызвало недовольство у некоторых разработчиков в Microsoft; они прекратили обновление Некоторые сотрудники Netscape вместе основали Mozilla, переписали ядро ​​на основе ядра Mosaic и разработали Gecko. Но на самом деле браузер ядра Gecko по-прежнему Firefox (Firefox) с большинством пользователей, поэтому его иногда называют ядром Firefox. Кроме того, Gecko также является кроссплатформенным ядром, которое можно использовать в Windows, BSD, Linux и Mac OS X. к  
Дополнение: движком JavaScript является SpiderMonkey. к  
Общие браузеры с ядром Gecko: [1] Mozilla Firefox, Mozilla SeaMonkey, waterfox (64-разрядная версия Firefox с открытым исходным кодом), Iceweasel, Epiphany (ранняя версия), Flock (ранняя версия), K-Meleon

Presto

Presto (pre-Opera kernel) (устарело): ядро, используемое в Opera 12.17 и более ранних версиях, прекратило разработку и теперь устарело. Ядро впервые использовалось в Opera7 в 2003 году. Особенностью этого движка является скорость рендеринга. Оптимизация достигла предела, но цена за счет веб-совместимости. к  
На самом деле это динамическое ядро. Самым большим отличием от предыдущих ядер является обработка скриптов. У Presto есть естественное преимущество. Вся или часть страницы может реагировать на события скрипта. Время от времени пересматривается. Кроме того, ядро ​​имеет самую высокую скорость при выполнении Javascrīpt. Согласно тестам в тех же условиях, время, необходимое для того, чтобы ядро ​​Presto выполнило тот же Javascrīpt, составляет всего около 1/3 от ядер Trident и Gecko (ядро Trident - самое медленное, но два Между ними нет большой разницы.) Один из модификаторов этой статьи считает, что приведенная выше информация о тестировании слишком старая и неполная, потому что небольшой тест, который он провел, показывает, что часть Presto является быстрой, а часть - медленной, а ядра в целом эквивалентны. В этом тесте, поскольку аппаратные условия компьютера Apple отличались от обычных компьютеров, ядро ​​WebCore не тестировалось. Жаль, что Presto - это коммерческий движок. Помимо Opera единственными, которые используют Presto, являются NDSBrowser, Wii Internet Channle, веб-браузер Nokia 770 и т. Д. Это в значительной степени ограничивает разработку Presto. к  
Opera перешла на ядро ​​Google Chrome Blink.

Webkit

Webkit (ядро Safari, прототип ядра Chrome, открытый исходный код): это собственное ядро ​​Apple и ядро, используемое браузером Safari от Apple. Движок Webkit включает в себя механизм верстки WebCore и механизм синтаксического анализа JavaScriptCore, оба производные от KDE KHTML и KJS, все они являются свободным программным обеспечением, лицензированным по соглашению GPL, и поддерживают разработку систем BSD. Так что Webkit - это также бесплатное программное обеспечение и открытый исходный код. С точки зрения безопасности, он не ограничен IE и Firefox, поэтому браузер Safari все еще очень безопасен в Китае. к  
ограничивается тем, что Mac OS X не используется широко, а браузер Safari когда-то был эксклюзивным браузером Mac OS X. Можно сказать, что само это ядро ​​имеет небольшую рыночную область применения, но, похоже, оно основано на новейшем браузере Опрос показывает, что рынок браузеров даже превзошел Presto. Конечно, это связано с растущей популярностью Apple после перехода на архитектуру x86, а также с тем, что Safari 3 наконец-то запустил версию для Windows. Есть также популярные браузеры, такие как OmniWeb и Shiira под Mac. к  
В быстродействующем режиме высокоскоростного браузера Sogou также используется Webkit в качестве ядра (с точки зрения понимания сценариев Chrome использует собственный разработанный движок V8). Ядро WebKit также широко используется в мобильных телефонах, например, мобильный телефон Google Gphone, iPhone Apple, браузер Nokia Series 60 и другие ядра ядра браузера - все основано на WebKit. к  
Основные браузеры WebKit: браузер Maxthon 3, [1] Apple Safari (Win / Mac / iPhone / iPad), мобильный браузер Symbian, браузер Android по умолчанию,

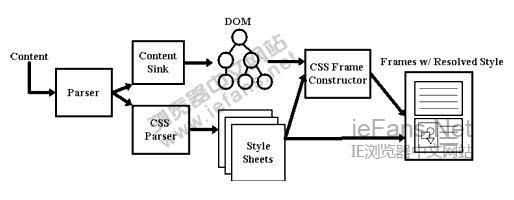
Blink

Внедрение продукции  
Blink - это механизм верстки браузера, разработанный Google и Opera Software. Google планирует использовать этот механизм рендеринга в рамках проекта Chromium, и анонсировал эту новость в апреле 2013 года. , Этот движок рендеринга является ветвью компонента WebCore движка с открытым исходным кодом WebKit и используется в браузерах Chrome (28 и более поздние версии), Opera (15 и более поздние версии) и Яндекса.

подводить итоги

IE (до 9) = Trident + Jscript  
IE (после 9) = трезубец + чакра  
360=Trident+Blink   
Sogou = Trident + WebKit  
Firefox=Gecko+JagerMonkey   
Safari=webkit=webcore+jscore   
Chrome=Blink+v8   
Opera=Blink+v8

Принцип реализации ядра браузера

Процесс рендеринга выглядит следующим образом:  


7) Что такое API

API (Application Programming Interface — программный интерфейс приложения, или интерфейс программирования приложений) — специальный протокол для взаимодействия компьютерных программ, который позволяет использовать функции одного приложения внутри другого.

— это код, который позволяет двум приложениям обмениваться данными с сервера. На русском языке его принято называть программным интерфейсом приложения. REST (Representational State Transfer) — это способ создания API с помощью протокола HTTP.

8. Что такое ендпоинты

Endpoint - это конечное защищаемое устройство. Компьютер с корпоративным антивирусом или мобилка с ним же. (в переводе с англ. — конечная точка) представляет собой некий шлюз, который соединяет серверные процессы приложения с внешним интерфейсом. Простыми словами, это адрес, на который отправляются сообщения.

Эндпоинты работают вместе с методами API. Это определенные URL-адреса, которые приложение использует для связи со сторонними службами и своими пользователями. URL — это роут, который может обрабатывать разные методы запроса. Каждый тип запроса — это отдельный эндпоинт.

9) URL (URI, URL, URN)

URL - Uniform Resource Locator (унифицированный определитель местонахождения ресурса)

URN - Unifrorm Resource Name (унифицированное имя ресурса)

URI - Uniform Resource Identifier (унифицированный идентификатор ресурса)



10) Идемпотентные HTTP методы

Метод HTTP является идемпотентным, если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера. Другими словами, идемпотентный метод не должен иметь никаких побочных эффектов (side-effects), кроме сбора статистики или подобных операций. Корректно реализованные методы [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD), [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) и [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) идемпотентны, но не метод [POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST). Также все [безопасные](https://developer.mozilla.org/ru/docs/Glossary/safe) методы являются идемпотентными.

Для идемпотентности нужно рассматривать только изменение фактического внутреннего состояния сервера, а возвращаемые запросами коды статуса могут отличаться: первый вызов [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) вернёт код [200](https://developer.mozilla.org/ru/docs/Web/HTTP/Status/200), в то время как последующие вызовы вернут код [404](https://developer.mozilla.org/ru/docs/Web/HTTP/Status/404). Из идемпотентности [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) неявно следует, что разработчики не должны использовать метод [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) при реализации RESTful API с функциональностью удалить последнюю запись.

11) Безопасные HTTP методы

Метод HTTP является безопасным, если он не меняет состояние сервера. Другими словами, безопасный метод проводит операции "только чтение" (read-only). Несколько следующих методов HTTP безопасные: [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD) или [OPTIONS](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/OPTIONS). Все безопасные методы являются также [идемпотентными](https://developer.mozilla.org/ru/docs/Glossary/Idempotent), как и некоторые другие, но при этом небезопасные, такие как [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) или [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE).

Даже если безопасные методы являются по существу "только для чтения", сервер всё равно может сменить своё состояние: например, он может сохранять статистику. Что существенно, так то, когда клиент вызывает безопасный метод, то он не запрашивает никаких изменений на сервере, и поэтому не создаёт дополнительную нагрузку на сервер. Браузеры могут вызывать безопасные методы, не опасаясь причинить вред серверу: это позволяет им выполнять некоторые действия, например, предварительная загрузка без риска. Поисковые роботы также полагаются на вызовы безопасных методов.

Безопасные методы не обязательно должны обрабатывать только статичные файлы; сервер может генерировать ответ "на-лету", пока скрипт, генерирующий ответ, гарантирует безопасность: он не должен вызывать внешних эффектов, таких как формирование заказов, отправка писем и др..

Правильная реализация безопасного метода - это ответственность серверного приложения, потому что сам веб-сервер, будь то Apache, nginx, IIS это соблюсти не сможет. В частности, приложение не должно разрешать изменение состояния сервера запросами [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET).

Вызов безопасного метода, не меняющего состояния сервера:

GET /pageX.html HTTP/1.1

Вызов небезопасного метода, который может поменять состояние сервера:

POST /pageX.html HTTP/1.1

12) Иденфикация, Аутентификация, Авторизация

Идентификация — процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно определяющий этого субъекта в информационной системе.

Аутентификация — процедура проверки подлинности, например проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.

Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий.

13) Что такое IP

IP-адрес — уникальный сетевой адрес узла в компьютерной сети, построенной на основе стека протоколов TCP/IP (TCP/IP – это набор интернет-протоколов, о котором мы поговорим в дальнейших статьях). IP-адрес представляет собой серию из 32 двоичных бит (единиц и нулей). Так как человек невосприимчив к большому однородному ряду чисел, такому как этот 11100010101000100010101110011110 (здесь, к слову, 32 бита информации, так как 32 числа в двоичной системе), было решено разделить ряд на четыре 8-битных байта и получилась следующая последовательность: 11100010.10100010.00101011.10011110. Это не сильно облегчило жизнь и было решение перевести данную последовательность в, привычную нам, последовательность из четырёх чисел в десятичной системе, то есть 226.162.43.158. 4 разряда также называются октетами. Данный IP адрес определяется протоколом IPv4. По такой схеме адресации можно создать более 4 миллиардов IP-адресов.

14) Что такое октаты в DNS

DNS (Domain Name Server) ─ серверы имен

Основная задача DNS — преобразование имен компьютеров в IPадреса и наоборот.

Для реализации системы DNS был создан специальный сетевой протокол DNS. В сети имеются специальные выделенные информационнопоисковые серверы - DNS-серверы.

Пространство имен DNS имеет вид дерева доменов с полномочиями, возрастающими по мере приближения к корню дерева.

Октет представляет собой восьмизначное двоичное число. В октете нулевое число будет 00000000, а 255 будет 11111111, максимальное число, которое может достигнуть октет.

Поскольку длинные последовательности из единиц и нулей трудно запомнить, IP адреса обычно записывают в десятичной форме. Для этого каждый октет адреса представляется в виде десятичного числа. Между собой октеты отделяются точкой. Иногда октеты обозначаются как w.x.y.z и называются “z-октет”, “у-октет”, “х-октет” и “w-октет”.

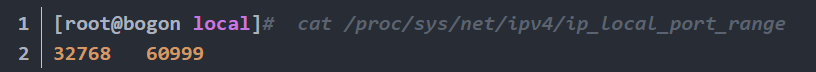
15) Что такое порт, сколько портов у Linux сервера

Порт сервера- это цифровой идентификатор (набор цифр до 16 Бит) программы, которая используется для создания определенного интернет соединения привязанного к IP- адресу. (механизм, с помощью которого операционная система определяет какой именно программе необходимо передать сетевой пакет. Здесь можно привести пример с домом. Например, почтальону необходимо доставить посылку. Он доставляет посылку к дому, это IP адрес компьютера. А дальше в самом доме уже должны разобраться в какую квартиру направить эту посылку. Номер квартиры - это уже порт.)

Каждое устройство имеющее выход в сеть имеет свой IP. Он нужен для того чтобы запрошенные данные были направлены именно на Ваш ПК. Порты служат для установления нескольких таких соединений. В свою очередь с помощью них сервер на котором расположен сайт создает соединение сервер―›ПК.

Данный канал служит для обмена данными между ПК и удаленным сервером.

Теоретически количество внешних портов для одной машины может достигать 65535, но фактическое количество подключений, которые могут быть установлены с внешним миром, по умолчанию составляет только 28232. Вы можете увидеть, выполнив следующие команды под Linux



То есть можно использовать порты из этого диапазона, поэтому один IP-адрес может отправлять не более 28232 TCP-запросов извне.  
Диапазон этого интервала можно временно изменить с помощью следующей команды, но после перезапуска системы он будет восстановлен до значения по умолчанию.



16) Уровни OSI



Первый, физический уровень (physical layer, L1)

Начнем с самого нижнего уровня. Он отвечает за обмен физическими сигналами между физическими устройствами, «железом». Компьютерное железо не понимает, что такое картинка или что на ней изображено, железу картинка понятна только в виде набора нулей и единиц, то есть бит. В данном случае бит является блоком данных протокола, сокращенно PDU (Protocol Data Unit).

Каждый уровень имеет свои PDU, представляемые в той форме, которая будет понятна на данном уровне и, возможно, на следующем до преобразования. Работа с чистыми данными происходит только на уровнях с пятого по седьмой.

Устройства физического уровня оперируют битами. Они передаются по проводам (например, через оптоволокно) или без проводов (например, через Bluetooth или IRDA, Wi-Fi, GSM, 4G и так далее).

Второй уровень, канальный (data link layer, L2)

Когда два пользователя находятся в одной сети, состоящей только из двух устройств — это идеальный случай. Но что если этих устройств больше?

Второй уровень решает проблему адресации при передаче информации. Канальный уровень получает биты и превращает их в кадры (frame, также «фреймы»). Задача здесь — сформировать кадры с адресом отправителя и получателя, после чего отправить их по сети.

У канального уровня есть два подуровня — это MAC и LLC. MAC (Media Access Control, контроль доступа к среде) отвечает за присвоение физических MAC-адресов, а LLC (Logical Link Control, контроль логической связи) занимается проверкой и исправлением данных, управляет их передачей.

На втором уровне OSI работают коммутаторы, их задача — передать сформированные кадры от одного устройства к другому, используя в качестве адресов только физические MAC-адреса.

Третий уровень, сетевой (network layer, L3)

На третьем уровне появляется новое понятие — маршрутизация. Для этой задачи были созданы устройства третьего уровня — маршрутизаторы (их еще называют роутерами). Маршрутизаторы получают MAC-адрес от коммутаторов с предыдущего уровня и занимаются построением маршрута от одного устройства к другому с учетом всех потенциальных неполадок в сети.

На сетевом уровне активно используется протокол ARP (Address Resolution Protocol — протокол определения адреса). С помощью него 64-битные MAC-адреса преобразуются в 32-битные IP-адреса и наоборот, тем самым обеспечивается инкапсуляция и декапсуляция данных.

Четвертый уровень, транспортный (transport layer, L4)

Все семь уровней модели OSI можно условно разделить на две группы:

Media layers (уровни среды),

Host layers (уровни хоста).

Уровни группы Media Layers (L1, L2, L3) занимаются передачей информации (по кабелю или беспроводной сети), используются сетевыми устройствами, такими как коммутаторы, маршрутизаторы и т.п. Уровни группы Host Layers (L4, L5, L6, L7) используются непосредственно на устройствах, будь то стационарные компьютеры или портативные мобильные устройства.

Четвертый уровень — это посредник между Host Layers и Media Layers, относящийся скорее к первым, чем к последним, его главной задачей является транспортировка пакетов. Естественно, при транспортировке возможны потери, но некоторые типы данных более чувствительны к потерям, чем другие. Например, если в тексте потеряются гласные, то будет сложно понять смысл, а если из видеопотока пропадет пара кадров, то это практически никак не скажется на конечном пользователе. Поэтому, при передаче данных, наиболее чувствительных к потерям на транспортном уровне используется протокол TCP, контролирующий целостность доставленной информации.

Для мультимедийных файлов небольшие потери не так важны, гораздо критичнее будет задержка. Для передачи таких данных, наиболее чувствительных к задержкам, используется протокол UDP, позволяющий организовать связь без установки соединения.

При передаче по протоколу TCP, данные делятся на сегменты. Сегмент — это часть пакета. Когда приходит пакет данных, который превышает пропускную способность сети, пакет делится на сегменты допустимого размера. Сегментация пакетов также требуется в ненадежных сетях, когда существует большая вероятность того, что большой пакет будет потерян или отправлен не тому адресату. При передаче данных по протоколу UDP, пакеты данных делятся уже на датаграммы. Датаграмма (datagram) — это тоже часть пакета, но ее нельзя путать с сегментом.

Главное отличие датаграмм в автономности. Каждая датаграмма содержит все необходимые заголовки, чтобы дойти до конечного адресата, поэтому они не зависят от сети, могут доставляться разными маршрутами и в разном порядке. Датаграмма и сегмент — это два PDU транспортного уровня модели OSI. При потере датаграмм или сегментов получаются «битые» куски данных, которые не получится корректно обработать.

Первые четыре уровня — специализация сетевых инженеров, но с последними тремя они не так часто сталкиваются, потому что пятым, шестым и седьмым занимаются разработчики.

Пятый уровень, сеансовый (session layer, L5)

Пятый уровень оперирует чистыми данными; помимо пятого, чистые данные используются также на шестом и седьмом уровне. Сеансовый уровень отвечает за поддержку сеанса или сессии связи. Пятый уровень оказывает услугу следующему: управляет взаимодействием между приложениями, открывает возможности синхронизации задач, завершения сеанса, обмена информации.

Службы сеансового уровня зачастую применяются в средах приложений, требующих удаленного вызова процедур, т.е. чтобы запрашивать выполнение действий на удаленных компьютерах или независимых системах на одном устройстве (при наличии нескольких ОС).

Примером работы пятого уровня может служить видеозвонок по сети. Во время видеосвязи необходимо, чтобы два потока данных (аудио и видео) шли синхронно. Когда к разговору двоих человек прибавится третий — получится уже конференция. Задача пятого уровня — сделать так, чтобы собеседники могли понять, кто сейчас говорит.

Шестой уровень, представления данных (presentation layer, L6)

О задачах уровня представления вновь говорит его название. Шестой уровень занимается тем, что представляет данные (которые все еще являются PDU) в понятном человеку и машине виде. Например, когда одно устройство умеет отображать текст только в кодировке ASCII, а другое только в UTF-8, перевод текста из одной кодировки в другую происходит на шестом уровне.

Шестой уровень также занимается представлением картинок (в JPEG, GIF и т.д.), а также видео-аудио (в MPEG, QuickTime). Помимо перечисленного, шестой уровень занимается шифрованием данных, когда при передаче их необходимо защитить.

Седьмой уровень, прикладной (application layer)

Седьмой уровень иногда еще называют уровень приложений, но чтобы не запутаться можно использовать оригинальное название — application layer. Прикладной уровень — это то, с чем взаимодействуют пользователи, своего рода графический интерфейс всей модели OSI, с другими он взаимодействует по минимуму.

Все услуги, получаемые седьмым уровнем от других, используются для доставки данных до пользователя. Протоколам седьмого уровня не требуется обеспечивать маршрутизацию или гарантировать доставку данных, когда об этом уже позаботились предыдущие шесть. Задача седьмого уровня — использовать свои протоколы, чтобы пользователь увидел данные в понятном ему виде.

Протоколы здесь используют UDP (например, DHCP) или TCP (например, HTTP, HTTPS, SFTP (Simple FTP), DNS). Прикладной уровень является самым верхним по иерархии, но при этом его легче всего объяснить.

17) Хедеры http запросов

