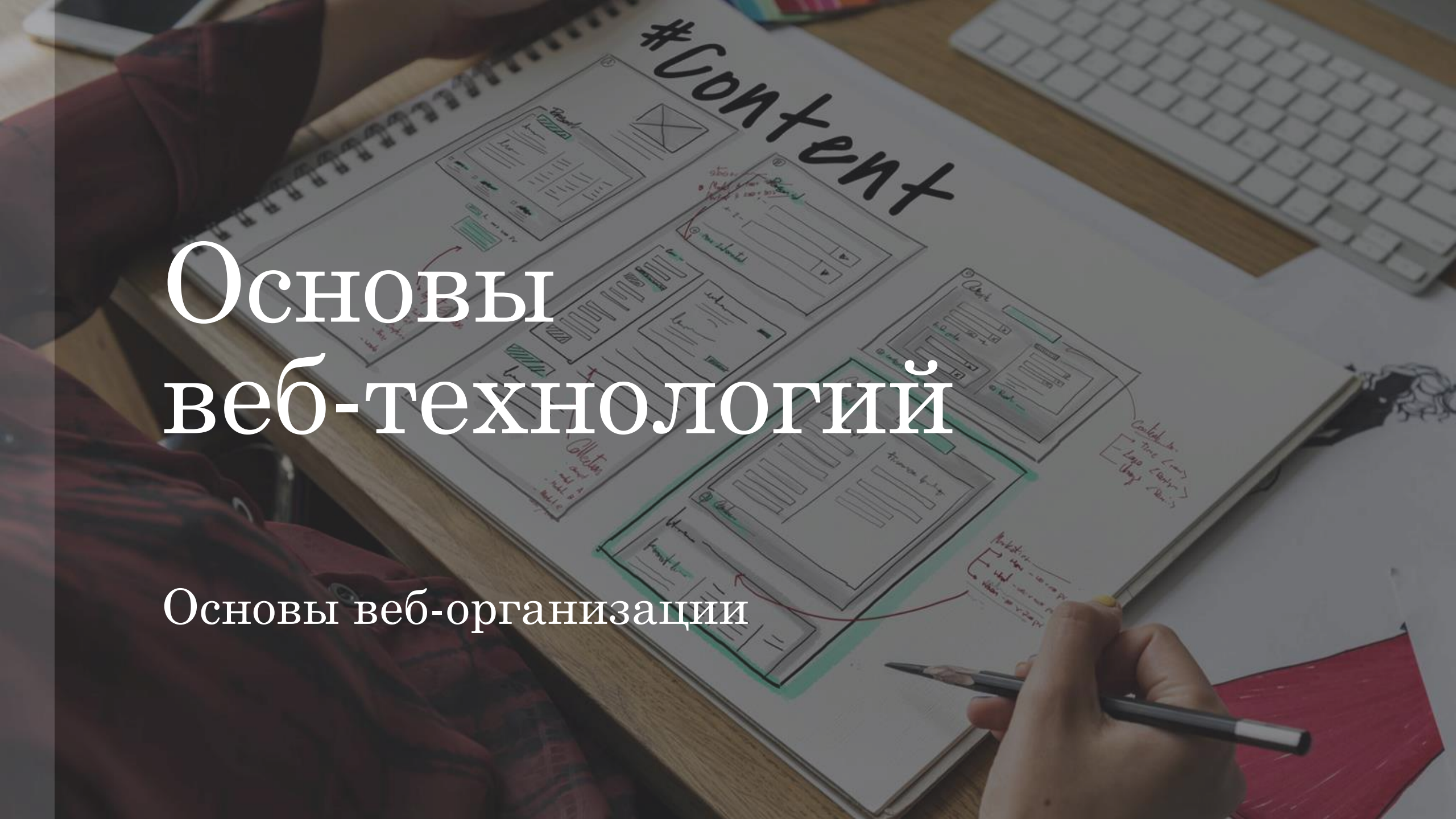


Основы веб-технологий

Основы веб-организации

Интернет и всемирная паутина

- **Интернет** – объединение компьютерных сетей, в котором ресурсы всех входящих в него компьютеров соединены в единое информационное пространство посредством телефонных линий, специальных сетевых кабелей (например, наиболее распространенный в наше время волоконно-оптический кабель), беспроводных устройств передающих и принимающих устройств (например, спутниковая связь) или любых других электронных средств связи.

  Сеть состоит из специализированных компьютеров – **серверов**, на которых аккумулируются большие объемы разного рода информационных ресурсов (документы, аудио-, видео-, анимационные файлы, базы данных и т.п.), и **клиентских компьютеров** (компьютер-клиент), через которые осуществляется вход в сеть. Все виды информационных ресурсов доступны всем компьютерам, находящимся в сети, в равной степени.

  Подключение компьютера к Интернету (выход в Интернет) обеспечивают специальные компании – интернет-провайдеры.

- **Всемирная паутина, глобальная сеть** (world wide web, www) – это собрание электронных документов (веб-страниц), соединенных гиперссылками (линками). Эти документы хранятся на специализированных компьютерах (веб-серверах), доступ к ним обеспечивается с помощью присваемого каждому документу уникального индивидуального интернет-адреса – URL(URI)).

  При введении в адресную строку программы просмотра веб-страниц (интернет-браузера) адреса веб-страницы или при активизации ссылки в Интернете клиентский компьютер посылает запрос на веб-сервер, на котором находится искомый файл, и в окно браузера на клиентском компьютере загружается его копия.

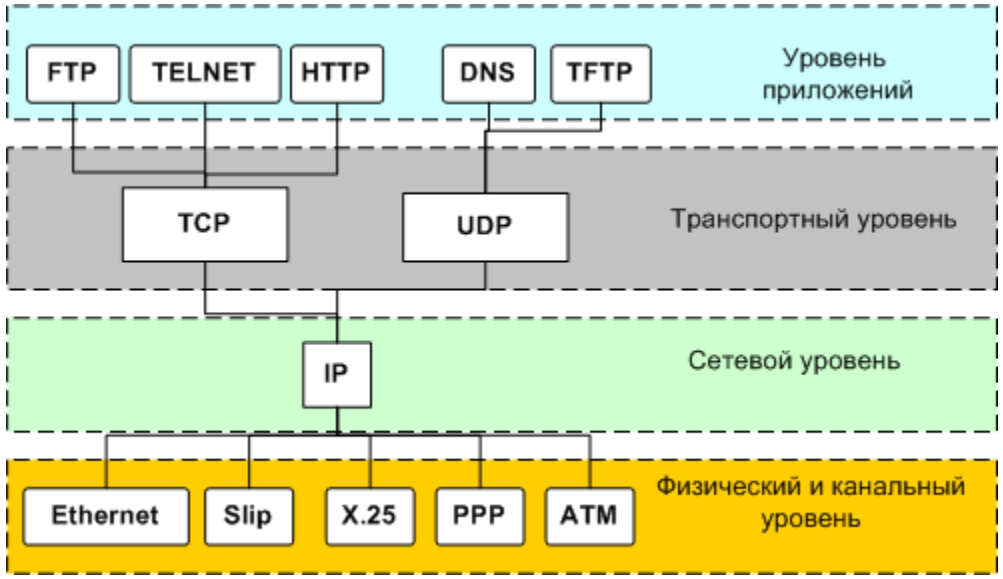
Основные термины Интернета

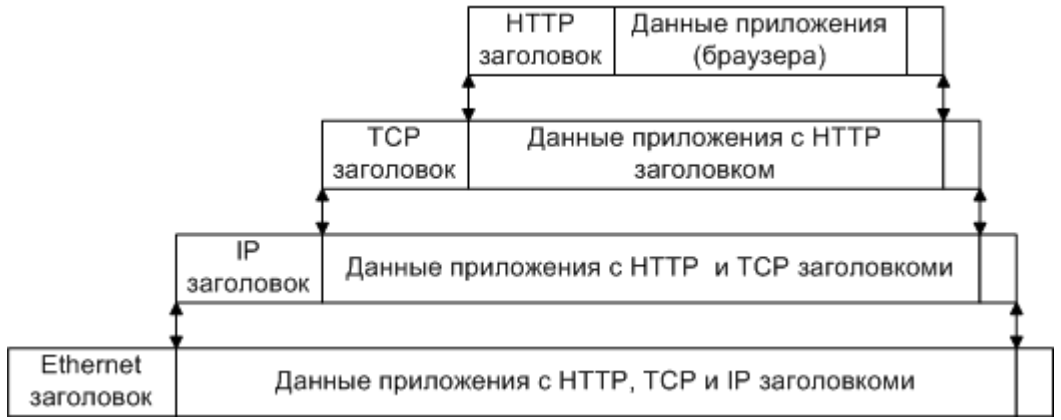
- **Гиперссылка, ссылка, линк** (англ. hyperlink, link) – обычно выделяемый цветом или с помощью подчеркивания какой-то элемент файла веб-страницы (текст, графический элемент и др.), который является исходным пунктом перехода на другие веб-страницы. Обычно гиперссылка может быть активизирована при наведении на нее курсора мыши, что отражается изменением цвета гиперссылки и (или) ее формы, а также изменением формы самого курсора. Часто при графическом обозначении гиперссылки рядом с ней (или в нижней части окна программы просмотра веб-страниц – веб-браузера) появляется текстовая подсказка об адресате этой ссылки или интернет-адрес соответствующей веб-страницы.
- **Веб-страница** (англ. web-page) – компьютерный файл, созданный с использованием специального языка гипертекстовой разметки HTML. Веб-страница может содержать не только текст, но и включать любую мультимедийную информацию (графику, аудио, видео, анимацию и т.п.), а также средства интерактивного взаимодействия с данным веб-сервером (формы и т.п.). Каждая веб-страница имеет свой уникальный интернет-адрес, который определяет ее местоположение на веб-сервере. На веб-странице могут быть ссылки на другие веб-страницы, которые оформляются с помощью специальной команды в языке HTML с указанием их интернет-адреса.
- **Веб-сайт** (англ. web-site) – собрание веб-страниц, которые объединены по какому-то принципу: тематически (например, материалы по истории русской литературы и т.п.), институционально (веб-сайт НИЯУ МИФИ) или персонально (веб-сайт отдельного физического или виртуального лица), функционально (коммерческие веб-сайты, интернет-СМИ и т.п.). Обычно веб-сайты имеют так называемую домашнюю страницу (англ. home-page), на которой в том или ином виде представлен каталог (гипертекстовый список) входящих в веб-сайт веб-страниц, обеспечивающий с помощью гиперссылок эффективную навигацию в информационном пространстве данного веб-сайта.

TCP/IP

Основополагающим протоколом сети Internet является протокол TCP/IP. TCP/IP это два различных протокола, тесно связанных между собой. TCP (Transmission Control Protocol) — протокол управления передачей. Он определяет, каким образом информация должна быть разбита на пакеты и отправлена по каналам связи. TCP располагает пакеты в нужном порядке, а также проверяет каждый пакет на наличие ошибок при передаче. Каждый информационный пакет содержит IP-адреса (IP – Internet Protocol) компьютера-отправителя и компьютера-получателя. Специальные компьютеры, называемые маршрутизаторами, используя IP-адреса, направляют информационные пакеты в нужную сторону, то есть к указанному в них получателю.

В модели TCP/IP предполагается прохождение информации через четыре уровня:





Инкапсуляция пакетов в стеке TCP/IP

Адресация в Интернете

Компьютер в сети TCP/IP может иметь адреса трех уровней (но не менее двух)::

- Физический. Локальный адрес узла, определяемый технологией, с помощью которой построена отдельная сеть, в которую входит данный узел. Для узлов, входящих в локальные сети – это MAC-адрес сетевого адаптера или порта маршрутизатора. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизовано.

- Сетевой. IP-адрес, состоящий из 32 бит (IPv4) или 128 бит (IPv6). Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet NIC (Network Information Center), если сеть должна работать как составная часть Internet. Обычно провайдеры услуг Internet получают диапазоны адресов у подразделений INIC, а затем распределяют их между своими абонентами. Номер узла в протоколе IP назначается независимо от локального адреса узла. Деление IP-адреса на поле номера сети и номера узла – гибкое, и граница между этими полями может устанавливаться весьма произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

- Символьный идентификатор-имя. Такой адрес, называемый также DNS-именем, используется на прикладном уровне.

IPv4

IPv4 адрес состоит из 4 блоков по 8 бит, каждый из которых называется октетом, и записывается в виде десятичных цифр, разделенных точкой. Значение каждого из четырех октетов находится в диапазоне от 0 до 255. Всего длина такого адреса 32 бита. Что на самом деле превращается в 4,3 миллиарда комбинаций.

IP адрес делится на две части: идентификатор сети и узла (хоста)

Маска подсети

В протоколе TCP/IP части IP-адреса, которые используются в качестве сетевых адресов и адресов хоста, не выделяются. Если у вас нет дополнительных сведений, то сетевые адреса и адреса хоста не могут быть определены. Эти сведения предоставляются в другом 32-битовом номере, который называется маской подсети.

Пример: IP-адрес 192.168.123.132, маска подсети — 255.255.255.0.

255 в двоичном представлении равно 11111111. Таким образом, маска подсети будет иметь вид 11111111.11111111.11111111.00000000.

Если выстроить IP-адрес и маску подсети вместе, можно разделить сетевую часть адреса сети и адрес хоста:

110000000.10101000.01111011.10000100 — IP-адрес (192.168.123.132)
11111111.11111111.11111111.00000000 — маска подсети (255.255.255.0)

Первые 24 бита (количество единиц в маске подсети) идентифицируются как адрес сети. Последние 8 битов (количество оставшихся нулей в маске подсети) идентифицируются как адрес узла. Таким образом, получаются следующие адреса:

11000000.10101000.01111011.00000000 — адрес сети (192.168.123.0)
00000000.00000000.00000000.10000100 — адрес узла (000.000.000.132)

Почти все десятичные маски подсети преобразовываются в двоичные числа, представленные единицами слева и нолями справа.

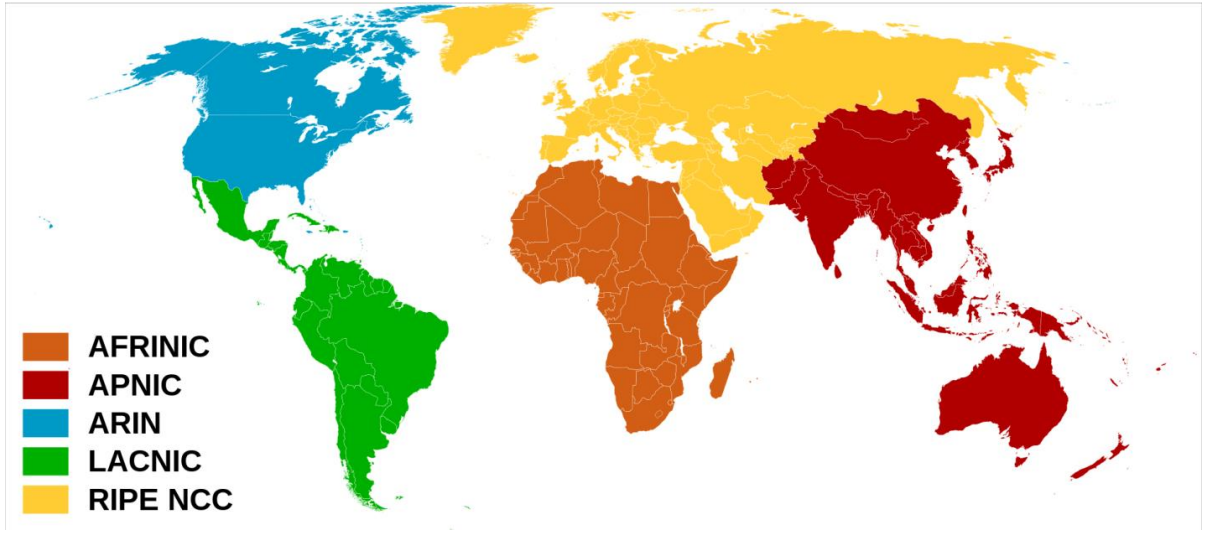
Возможные значения масок подсети при методе адресации

Всего адресов	битов	Префикс	Десятичная маска
1	0	/32	255.255.255.255
2	1	/31	255.255.255.254
4	2	/30	255.255.255.252
8	3	/29	255.255.255.248
16	4	/28	255.255.255.240
32	5	/27	255.255.255.224
64	6	/26	255.255.255.192
128	7	/25	255.255.255.128
256	8	/24	255.255.255.0
512	9	/23	255.255.254.0
1024	10	/22	255.255.252.0
2048	11	/21	255.255.248.0
4096	12	/20	255.255.240.0
8192	13	/19	255.255.224.0
16384	14	/18	255.255.192.0
32768	15	/17	255.255.128.0

Всего адресов	битов	Префикс	Десятичная маска
65536	16	/16	255.255.0.0
131072	17	/15	255.254.0.0
262144	18	/14	255.252.0.0
524288	19	/13	255.248.0.0
1048576	20	/12	255.240.0.0
2097152	21	/11	255.224.0.0
4194304	22	/10	255.192.0.0
8388608	23	/9	255.128.0.0
16777216	24	/8	255.0.0.0
33554432	25	/7	254.0.0.0
67108864	26	/6	252.0.0.0
134217728	27	/5	248.0.0.0
268435456	28	/4	240.0.0.0
536870912	29	/3	224.0.0.0
1073741824	30	/2	192.0.0.0
2147483648	31	/1	128.0.0.0
4294967296	32	/0	0.0.0.0

Кто раздает IP адреса?

ICANN (Internet Corporation for Assigned Names and Numbers) – Интернет-корпорация по присвоению имен и номеров, некоммерческая организация, созданная в 1998 году для регулирования вопросов, связанных с доменными именами, IP-адресами и прочими аспектами функционирования Интернета, включая стабильность и безопасность. ICANN осуществляет координацию и поддержку функционирования системы имен и адресов в Интернете. Одной из основных политик ICANN является расширение адресного пространства за счет создания новых доменов верхнего уровня, в том числе с использованием символов национальных алфавитов.



RIPE NCC, в зону которого входит Россия, исчерпал адреса 25 ноября 2019 года

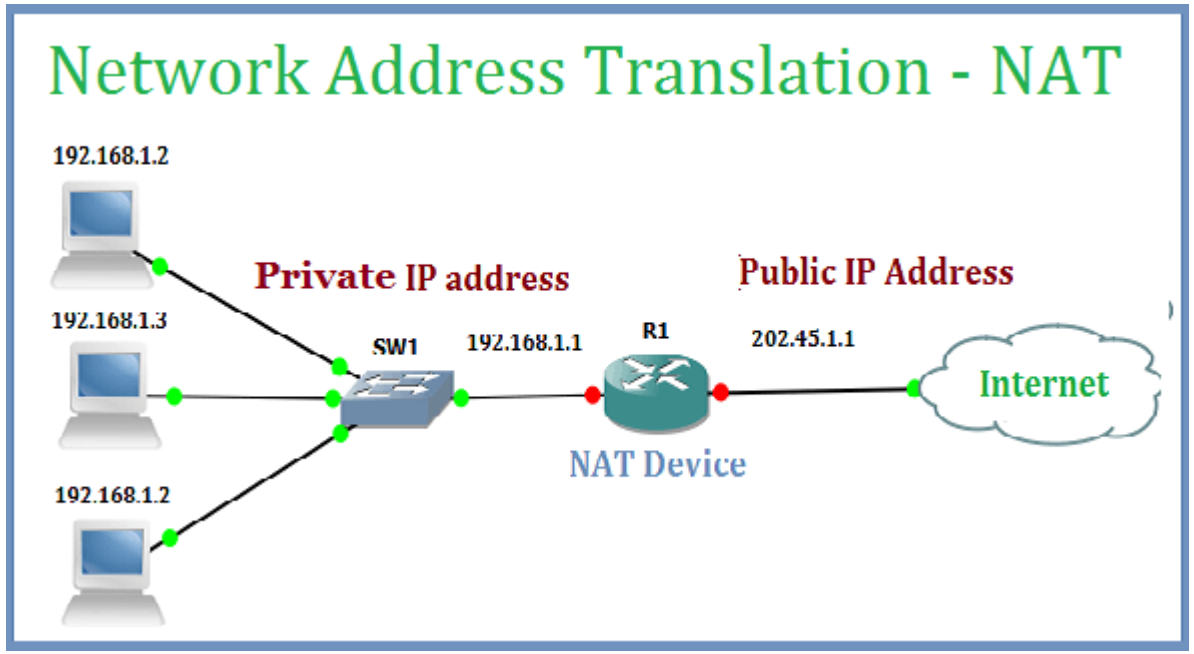
NAT (Network Address Translation)

NAT (Network Address Translation) – технология преобразовывания приватных IP-адресов во внешние в IPv4.

В частной сети используются приватные (серые) IP-адреса, которые не используются в Интернете. Группой проектирования Интернета в 1994 были выбраны следующие подсети (что остаётся актуальным и по сей день):

- 10.0.0.0 — 10.255.255.255 (маска подсети: 255.0.0.0 или /8, 16 777 216 хостов)
- 172.16.0.0 — 172.31.255.255 (маска подсети: 255.240.0.0 или /12, 1 048 576 хостов)
- 192.168.0.0 — 192.168.255.255 (маска подсети: 255.255.0.0 или /16, 65 536 хостов)

Также для петлевых интерфейсов (не используется для обмена между узлами сети) зарезервирован диапазон 127.0.0.0 — 127.255.255.255 (маска подсети: 255.0.0.0 или /8).



IPv6

В 1996 году была разработана и одобрена еще более расширяемая и масштабируемая версия технологии IP — IP версии 6 (IPv6). Протокол IPv6 использует для адресации 128 битов вместо 32-х битов в IPv4. В стандарте IPv6 используется шестнадцатеричная запись числа для представления 128-битовых адресов.

$$\mathbf{2^{128} = 340\ 282\ 366\ 920\ 938\ 463\ 463\ 374\ 607\ 431\ 768\ 211\ 456 \approx 3.4*10^{38}}$$

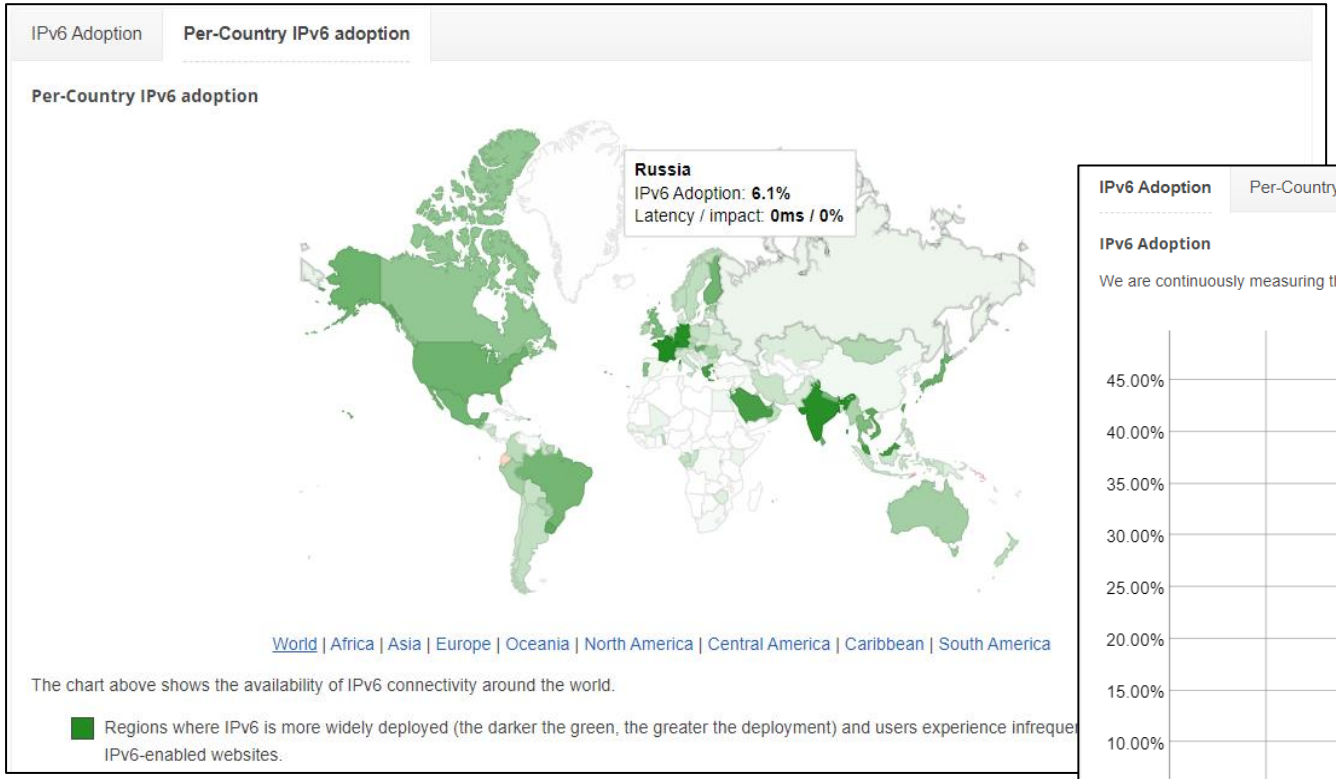
Пример адреса IPv6

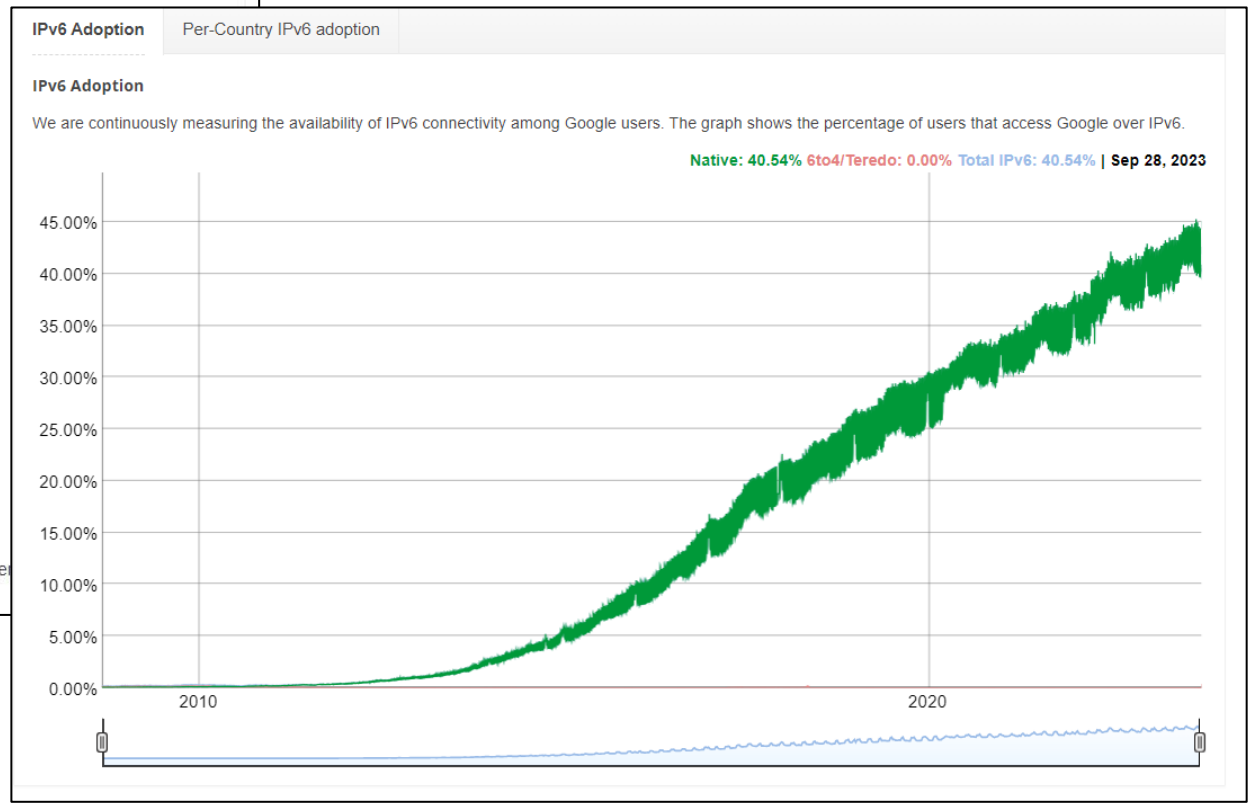
2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d

Частные диапазоны IP-адресов: fc00::/7 — 7-битный префикс адреса.

IPv6

Google стал активно использовать IPv6 еще в 2008 году. Официальный всемирный запуск IPv6 состоялся в 2012. Что же сейчас с переходом на IPv6?





Система доменных имен (Domain Name System, DNS)

DNS (англ.) — распределённая система (распределённая база данных), способная по запросу, содержащему доменное имя хоста (компьютера или другого сетевого устройства), сообщить IP адрес или (в зависимости от запроса) другую информацию. DNS работает в сетях TCP/IP. Как частный случай, DNS может хранить и обрабатывать и обратные запросы, определения имени хоста по его IP адресу — IP адрес по таблице соответствия преобразуется в доменное имя, и посылается запрос на информацию типа «PTR» (Reverse DNS).

Какие DNS-серверы бывают

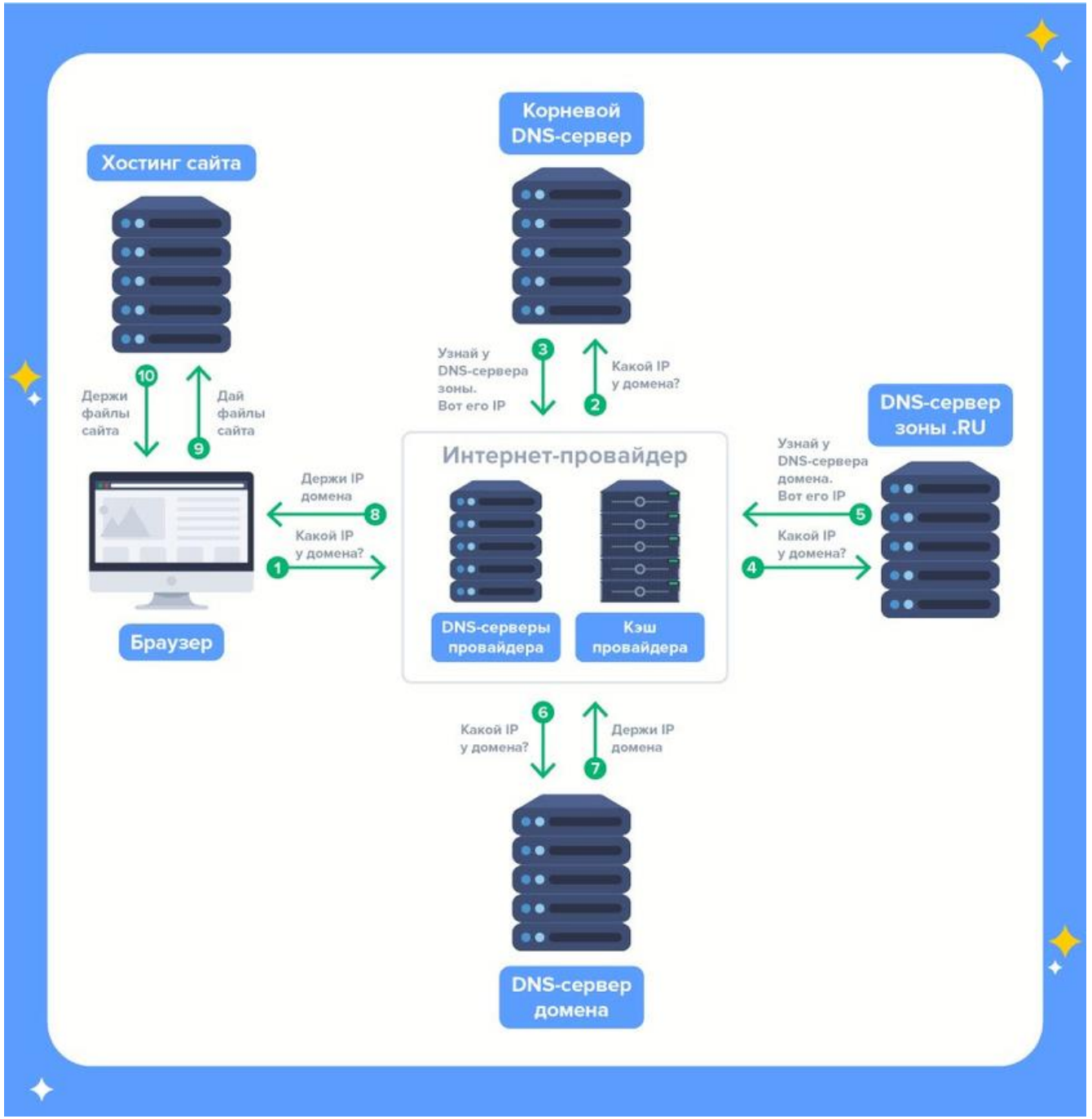
1. Корневые серверы. Их всего 13. Они принадлежат разным операторам. Официальная информация, где они находятся и кому принадлежат, публикуется на сайте Ассоциации операторов корневых серверов DNS. Большинство из них расположены в США, несколько - в Европе и Японии. Для устойчивости системы у каждого корневого сервера есть копии в разных странах. Управлением DNS-серверов занимается международная некоммерческая организация ICANN, расположенная в США.

2. DNS-серверы доменных зон (TLD-серверы, TLD – Top Level Domain). Например, серверы зоны .RU, .ART, .SITE. DNS-серверы зоны хранят только информацию о DNS-серверах всех доменов в этой зоне, IP самого сайта они не знают.

3. Серверы самих доменов (авторитативные DNS-серверы). Эти серверы знают IP-адреса конкретных доменов. Именно они могут отправить к хостингу, где лежат файлы сайта.

4. Локальные, или кэширующие DNS. Это серверы интернет-провайдеров. На них сохраняются данные с предыдущих запросов пользователей. Помогают связаться с другими видами DNS-серверов.

Как работает система доменных имён



DNS

Информация о домене

Многие домены верхнего уровня поддерживают сервис whois (https://whois.ru/), который позволяет узнать кому делегирован домен, и другую техническую информацию.

Регистрация домена

Регистрация домена — процедура получения доменного имени. Заключается в создании записей, указывающих на администратора домена, в базе данных DNS. Порядок регистрации и требования зависят от выбранной доменной зоны. Регистрация домена может быть выполнена как организацией-регистратором, так и частным лицом, если это позволяют правила выбранной доменной зоны.

Зарезервированные доменные имена

Документ RFC 2606 (Reserved Top Level DNS Names — Зарезервированные имена доменов верхнего уровня) определяет названия доменов, которые следует использовать в качестве примеров (например, в документации), а также для тестирования. Кроме example.com, example.org и example.net, в эту группу также входят test, invalid и др.

Сетевые порты

Сетевой порт – это некое виртуальное расширение, дополнение к сетевому адресу (IP-адресу).

Сетевой порт с технической точки зрения – это натуральное число от 0 до 65535, которое записывается в заголовках протоколов транспортного уровня сетевой модели OSI. Оно используется для определения программы или процесса-получателя пакета в пределах одного IP-адреса.

Основным предназначением портов является прием и передача данных определенного вида, а также устранение ошибки неоднозначности при попытке установить связь с хостом по IP-адресу. Для обеспечения трансляции данных с веб-сервера необходимо указать IP адрес хоста и номер порта, определяющий программу веб-сервера.

Для передачи данных используются транспортные протоколы, самые популярные – это TCP/IP (Transmission Control Protocol/Internet Protocol) и UDP (User Datagram Protocol). Обычно приложение либо ожидает входящие данные (или запроса на соединение), либо посылает данные (или запрос на соединение) на известный порт, открытый приложением-сервером (роль клиента).

По умолчанию приложению выдается порт с произвольным номером (к примеру, ближайшим свободным, большим 1234). При необходимости приложение может запросить конкретный (предопределённый) номер порта. Так веб-серверы обычно открывают для ожидания соединения предопределённый порт 80 протокола TCP.

Порты TCP не пересекаются с портами UDP. То есть, порт 1234 протокола TCP не будет мешать обмену по UDP через порт 1234.

Сетевые порты

- В компьютере точное количество портов – 65 535. И ух них есть своя градация. То есть порты с номерами до 1023 Линукс и Unix-подобными ОС считаются за «критически важные» для сетевой деятельности системы, так что для доступа к ним и службам, с ними связанными часто требуются root права. Windows также их считает системными и пристально следит за ними.

- Порты от 1024 до 49151 имеют статус «готовые к регистрации». Это значит, что эти порты зарезервированы или могут быть зарезервированы за определёнными службами. Однако стоит отметить, что они за этими сервисами не закреплены прочными правилами, но могут дать ключ для распознавания запущенной программы на стороне хоста.

- Остальные (начиная с 49152) порты не зарегистрированы и используются по усмотрению пользователей ОС и имеют название – «динамические» порты. Так что запоминать, какой порт для какой службы установлен, просто бесполезно.

- Основные сетевые порты:

  - 21 – ftp;
  - 22 – ssh;
  - 23 – telnet;
  - 25 – smtp;
  - 43 – whois;
  - 53 – dns;
  - 68 – dhcp;
  - **80 – http;**
  - 110 – pop3;
  - 115 – sftp;
  - 119 – nntp;
  - 123 – ntp;
  - 139 – netbios;
  - 143 – imap;
  - 161 – snmp;
  - 179 – bgp;
  - 220 – imap3;
  - 389 – ldap;
  - **443 – https;**
  - 993 – imaps;
  - 1723 – pptp;
  - 2049 – nfs;
  - 3306 – mysql;
  - 3389 – rdp;
  - 5060 – sip;
  - **8080 – http** (альтернативный)

URI, URL, URN

- **URL** – Uniform Resource Locator (унифицированный определитель местонахождения ресурса): Исторически возник самым первым из понятий и закрепился как синоним термина веб-адрес. URL определяет местонахождение ресурса в сети и способ его (ресурса) извлечения. Это позволяет нам полностью узнать: как, кому и где можно достать требуемый ресурс, вводя понятия схемы, данных авторизации и местонахождения.

- **URN** – Uniform Resource Name (унифицированное имя ресурса): Неизменяемая последовательность символов определяющая только имя некоторого ресурса. Смысл URN в том, что им единоразово и уникально именуется какая-либо сущность в рамках конкретного пространства имен (контекста), либо без пространства имен, в общем (что не желательно). Таким образом, URN способен преодолеть недостаток URL связанный с возможным будущим изменением и перемещением ссылок, однако, теперь для того, чтобы знать местонахождение URN ресурса необходимо обращаться к системе разрешения имен URN, в которой он должен быть зарегистрирован.

- **URI** – Uniform Resource Identifier (унифицированный идентификатор ресурса): это лишь обобщенное понятие (множество) идентификации ресурса, включающее в нашем случае как URL, так и URN, как по отдельности, так и совместно.

URL — Uniform Resource Locator

URL — это идентификатор, частный случай URI, обозначающий адрес к какому-либо веб-ресурсу. То, чем пользуемся в веб-браузере, и есть URL.

Синтаксис URL
схема://хост:порт/путь?параметры-запроса#фрагмент

Пример URL
https://home.mephi.ru/study_groups?level=1&organization_id=1&term_id=17

Схема
Чаще всего это **HTTPS** для защищённых соединений и **HTTP** для простых соединений.
Иерархическая часть
Обязательно состоит из адреса хоста, опционально порта и пути к ресурсу на сервере. Иерархическая часть может состоять только из адреса хоста с окончанием / с пустым адресом пути (например https://home.mephi.ru /).
Параметры запроса
В URL компонент запроса, идущий после символа ?, обычно состоит из пар имя=значение обозначающих дополнительные параметры запроса.

Фрагмент
Веб-браузеры, по умолчанию, переходят к отображению элемента HTML со значением атрибута id из компонента фрагмента после символа #.

URN — Uniform Resource Name

URN обозначает ресурс со статичным именем без привязки к расположению этого ресурса. Если **URL**, при перемещении объекта, становится ошибочным, то **URN** всегда ссылается на ресурс по его уникальному имени, в независимости от места размещения этого ресурса.

Синтаксис URN
схема:NID:NSS

Пример URN
urn:uuid:d5aa821e-52bb-4f7e-a782-7662ed5eaf5c

NID
Компонент URN адреса обозначающий пространство имён идентификатора.
NSS
Следующий компонент содержит конкретный идентификатор из пространства имён указанном ранее.

URN — это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение. Например, URN urn:ISBN:0-395-36341-1 — это URI, который указывает на ресурс (книгу) 0-395-36341-1 в пространстве имён ISBN, но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить или на каком сайте скачать.

URI — Uniform Resource Identifier

URI — это основной идентификатор для имён всех ресурсов размещённых в WWW, состоящий из последовательности символов указывающих на виртуальный или физический ресурс. Используется для идентификации ресурсов в сети интернет по их расположению или имени, а чаще и тем и тем. URI может являться **URL** или **URN**, или обоими сразу.

Синтаксис URI
схема:[//[пользователь:пароль@]хост[:порт]][/]путь[?запрос][#фрагмент]

Пример URI
http://site:dDfsd876@localhost:3306/new_db/sample_table?column=created_at#100

Схема
Первым компонентом идентификатора является обозначение схемы, чаще всего являющейся протоколом обмена (ftp, http или ssh).
Иерархическая часть
Следующий компонент содержит путь к ресурсу и, при необходимости, авторизационные данные для доступа. Может состоять только из пути к ресурсу или из имени хоста и пути к ресурсу на удалённом сервере.
Запрос
Необязательный компонент содержащий дополнительные параметры для запроса после символа ?.
Фрагмент
Ещё один необязательный компонент для обозначения конкретного фрагмента в рамках запрашиваемого ресурса после символа #.

URI ПРОТИВ URL

- В последнее время появилась тенденция говорить просто URI о любой строке-идентификаторе, без дальнейших уточнений. Так что, возможно, термины URL и URN скоро уйдут в прошлое.
- Поскольку URI не всегда указывает на то, как получить ресурс, в отличие от URL, а только идентифицирует его, это даёт возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через Интернет (например, личность, автомобиль, город и проч.).
- Фактически, URI представляет собой расширенный набор URL-адресов и нечто, называемое URN. Таким образом, мы можем с уверенностью заключить, что все URL являются URI. Однако обратное неверно.

Протокол HTTP

HTTP (HyperText Transfer Protocol, дословно — «протокол передачи гипертекста») представляет собой протокол прикладного уровня, используемый для доступа к ресурсам Всемирной Паутины.

Клиенты и серверы взаимодействуют, обмениваясь одиночными сообщениями (а не потоком данных). Сообщения, отправленные клиентом, обычно веб-браузером, называются **запросами**, а сообщения, отправленные сервером, называются **ответами**.

HTTP не имеет состояния, но имеет сессию: не существует связи между двумя запросами, которые последовательно выполняются по одному соединению. Но, хотя ядро HTTP не имеет состояния, куки позволяют использовать сессии с сохранением состояния.

HTTP и соединения

Соединение управляется на транспортном уровне и потому принципиально выходит за границы HTTP. HTTP полагается на стандарт TCP, являющийся основанным на соединениях, несмотря на то, что соединение не всегда требуется.

HTTP/1.0 открывал TCP-соединение для каждого обмена запросом/ответом, имея два важных недостатка: открытие соединения требует нескольких обменов сообщениями, и потому медленно, хотя становится более эффективным при отправке нескольких сообщений, или при регулярной отправке сообщений: тёплые соединения более эффективны, чем холодные.

Для смягчения этих недостатков, HTTP/1.1 предоставил конвейерную обработку (которую оказалось трудно реализовать) и устойчивые(постоянные) соединения: лежащее в основе TCP соединение можно частично контролировать через заголовок Connection. HTTP/2 сделал следующий шаг, добавив мультиплексирование сообщений через простое соединение, помогающее держать соединение тёплым и более эффективным.

HTTP-сообщения: запросы и ответы

HTTP/1.1 и более ранние HTTP сообщения человекочитаемые. В версии HTTP/2 эти сообщения встроены в новую бинарную структуру, фрейм, позволяющий оптимизации, такие как компрессия заголовков и мультиплексирование. Даже если часть оригинального HTTP сообщения отправлена в этой версии HTTP, семантика каждого сообщения не изменяется и клиент воссоздаёт (виртуально) оригинальный HTTP-запрос. Это также полезно для понимания HTTP/2 сообщений в формате HTTP/1.1.

Данные между клиентом и сервером в рамках работы протокола передаются с помощью HTTP-сообщений. Они бывают двух видов:

- **Запросы (HTTP Requests)** — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок.
- **Ответы (HTTP Responses)** — сообщения, которые сервер отправляет *в ответ* на клиентский запрос.

Само по себе сообщение представляет собой информацию в текстовом виде, записанную в несколько строчек.

HTTP-сообщения: запросы и ответы

Само по себе сообщение представляет собой информацию в текстовом виде, записанную в несколько строчек.

В целом, как запросы HTTP, так и ответы имеют следующую структуру:

1. *Стартовая строка (start line)* — используется для описания версии используемого протокола и другой информации — вроде запрашиваемого ресурса или кода ответа. Как можно понять из названия, ее содержимое занимает ровно одну строчку.
2. *HTTP-заголовки (HTTP Headers)* — несколько строчек текста в определенном формате, которые либо уточняют запрос, либо описывают содержимое *тела* сообщения.
3. Пустая строка, которая сообщает, что все метаданные для конкретного запроса или ответа были отправлены.
4. Опциональное *тело сообщения*, которое содержит данные, связанные с запросом, либо документ (например HTML-страницу), передаваемый в ответе.

Стартовая строка HTTP-запроса

Стартовая строка HTTP-запроса состоит из трех элементов:

1. *Метод HTTP-запроса* (method, реже используется термин verb). Обычно это короткое слово на английском, которое указывает, что конкретно нужно сделать с запрашиваемым ресурсом. Например, метод GET сообщает серверу, что пользователь хочет получить некоторые данные, а POST — что некоторые данные должны быть помещены на сервер.

2. Цель запроса. Представлена указателем ресурса URL, который состоит из протокола, доменного имени (или IP-адреса), пути к конкретному ресурсу на сервере. Дополнительно может содержать указание порта, несколько параметров HTTP-запроса и еще ряд опциональных элементов.

3. Версия используемого протокола (либо HTTP/1.1, либо HTTP/2), которая определяет структуру следующих за стартовой строкой данных.

В примере ниже стартовая строка указывает, что в качестве метода используется GET, обращение будет произведено к ресурсу /index.html, по версии протокола HTTP/1.1:

Стартовая строка HTTP-запроса: Метод

Методы позволяют указать конкретное действие, которое мы хотим, чтобы сервер выполнил, получив наш запрос. Так, некоторые методы позволяют браузеру (который в большинстве случаев является источником запросов от клиента) отправлять дополнительную информацию в теле запроса — например, заполненную форму или документ.

Метод	Описание
GET	Позволяет запросить некоторый конкретный ресурс. Дополнительные данные могут быть переданы через строку запроса (Query String) в составе URL (например ?param=value).О составляющих URL мы поговорим чуть позже.
POST	Позволяет отправить данные на сервер. Поддерживает отправку различных типов файлов, среди которых текст, PDF-документы и другие типы данных в двоичном виде. Обычно метод POST используется при отправке информации (например, заполненной формы логина) и загрузке данных на веб-сайт, таких как изображения и документы.
HEAD	Обычно сервер в ответ на запрос возвращает заголовок и тело, в котором содержится запрашиваемый ресурс. Данный метод при использовании его в запросе позволит получить только заголовки, которые сервер бы вернул при получении GET-запроса к тому же ресурсу. Запрос с использованием данного метода обычно производится для того, чтобы узнать размер запрашиваемого ресурса перед его загрузкой.
PUT	Используется для создания (размещения) новых ресурсов на сервере. Если на сервере данный метод разрешен без надлежащего контроля, то это может привести к серьезным проблемам безопасности.
DELETE	Позволяет удалить существующие ресурсы на сервере. Если использование данного метода настроено некорректно, то это может привести к атаке типа «Отказ в обслуживании» (Denial of Service, DoS) из-за удаления критически важных файлов сервера.
OPTIONS	Позволяет запросить информацию о сервере, в том числе информацию о допускаемых к использованию на сервере HTTP-методов.
PATCH	Позволяет внести частичные изменения в указанный ресурс по указанному расположению.

Стартовая строка HTTP-запроса: URL

Получение доступа к ресурсам по HTTP-протоколу осуществляется с помощью указателя URL. Host указывается в заголовках, здесь же указывается оставшиеся части URL :

Путь
Указывает на ресурс, к которому производится обращение. Если данное поле не указано, то сервер в большинстве случаев вернет указатель по умолчанию (например index.html).

Запрос
Необязательный компонент содержащий дополнительные параметры для запроса после символа ?.

Фрагмент
Ещё один необязательный компонент для обозначения конкретного фрагмента в рамках запрашиваемого ресурса после символа #.

Стартовая строка HTTP-запроса: Версии HTTP

HTTP/2 стал первым бинарным протоколом. Если сравнивать его с прошлой версией протокола, то здесь разработчики поменяли методы распределения данных на фрагменты и их отправку от сервера к пользователю и наоборот. Новая версия протокола позволяет серверам доставлять информацию, которую клиент пока что не запросил. Это было внедрено с той целью, чтобы сервер сразу же отправлял браузеру для отображения документов дополнительные файлы и избавлял его от необходимости анализировать страницу и самостоятельно запрашивать недостающие файлы.

Еще одно отличие http 2.0 от версии 1.1 – мультиплексирование запросов и ответов для решения проблемы блокировки начала строки, присущей HTTP 1.1. Еще в новом протоколе можно сжимать HTTP заголовки и вводить приоритеты для запросов.

Благодаря переходу с протокола HTTP 1.1 на протокол HTTP/2 веб-сайты стали работать значительно быстрее за счет нескольких факторов:

- возможность отправлять несколько запросов с помощью одного TCP-соединения;
- задавания клиентом важности потоков, то есть, приоритета одних ресурсов над другими;
- сокращения HTTP-заголовков;
- push-отправки от сервера к клиенту данных, которые еще не были запрошены.

HTTP-заголовки

HTTP-заголовок представляет собой строку формата «Имя-Заголовок:Значение», с двоеточием(:) в качестве разделителя. Название заголовка не учитывает регистр, то есть между Host и host, с точки зрения HTTP, нет никакой разницы. Однако в названиях заголовков принято начинать каждое новое слово с заглавной буквы. Структура значения зависит от конкретного заголовка. Несмотря на то, что заголовок вместе со значениями может быть достаточно длинным, занимает он всего одну строчку.

В запросах может передаваться большое число различных заголовков, но все их можно разделить на три категории:

1. **Общего назначения**, которые применяются ко всему сообщению целиком.
2. **Заголовки запроса** уточняют некоторую информацию о запросе, сообщая дополнительный контекст или ограничивая его некоторыми логическими условиями.
3. **Заголовки представления**, которые описывают формат данных сообщения и используемую кодировку. Добавляются к запросу только в тех случаях, когда с ним передается некоторое тело.

Основные HTTP-заголовки

Заголовок	Описание
Host	Используется для указания того, с какого конкретно хоста запрашивается ресурс. В качестве возможных значений могут использоваться как доменные имена, так и IP-адреса. На одном HTTP-сервере может быть размещено несколько различных веб-сайтов. Для обращения к какому-то конкретному требуется данный заголовок.
User-Agent	Заголовок используется для описания клиента, который запрашивает ресурс. Он содержит достаточно много информации о пользовательском окружении. Например, может указать, какой браузер используется в качестве клиента, его версию, а также операционную систему, на которой этот клиент работает.
Refer	Используется для указания того, откуда поступил текущий запрос.
Accept	Позволяет указать, какой тип медиафайлов принимает клиент. В данном заголовке могут быть указаны несколько типов, перечисленные через запятую (‘ , ‘). А для указания того, что клиент принимает любые типы, используется следующая последовательность — */*.
Cookie	Данный заголовок может содержать в себе одну или несколько пар «Куки-Значение» в формате cookie=value. Куки представляют собой небольшие фрагменты данных, которые хранятся как на стороне клиента, так и на сервере, и выступают в качестве идентификатора. Куки передаются вместе с запросом для поддержания доступа клиента к ресурсу. Помимо этого, куки могут использоваться и для других целей, таких как хранение пользовательских предпочтений на сайте и отслеживание клиентской сессии. Несколько куки в одном заголовке могут быть перечислены с помощью символа точка с запятой (‘ ; ‘), который используется как разделитель.
Authorization	Используется в качестве еще одного метода идентификации клиента на сервере. После успешной идентификации сервер возвращает токен, уникальный для каждого конкретного клиента. В отличие от куки, данный токен хранится исключительно на стороне клиента и отправляется клиентом только по запросу сервера. Существует несколько типов аутентификации, конкретный метод определяется тем веб-сервером или веб-приложением, к которому клиент обращается за ресурсом.

Тело запроса

Завершающая часть HTTP-запроса — это его тело. Не у каждого HTTP-метода предполагается наличие тела. Так, например, методам вроде GET, HEAD, DELETE, OPTIONS обычно не требуется тело. Некоторые виды запросов могут отправлять данные на сервер в теле запроса: самый распространенный из таких методов — POST.

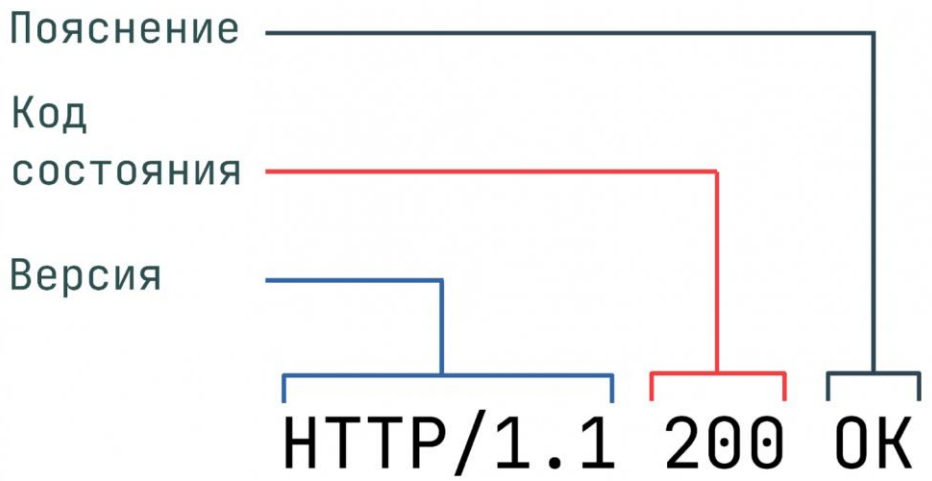
Пример HTTP-запроса

Ответы HTTP

HTTP-ответ является сообщением, которое сервер отправляет клиенту *в ответ* на его запрос. Его структура равна структуре HTTP-запроса: стартовая строка, заголовки и тело.

Стартовая строка HTTP-ответа называется **строкой статуса** (status line). На ней располагаются следующие элементы:

1. Уже известная нам по стартовой строке запроса *версия протокола* (HTTP/2 или HTTP/1.1).
2. *Код состояния*, который указывает, насколько успешно завершилась обработка запроса.
3. *Пояснение* — короткое текстовое описание к коду состояния. Используется исключительно для того, чтобы упростить понимание и восприятие человека при просмотре ответа.



Коды состояния и текст статуса

Коды состояния HTTP используются для того, чтобы сообщить клиенту статус их запроса. HTTP-сервер может вернуть код, принадлежащий одной из пяти категорий кодов состояния:

Категория	Описание
1xx	Коды из данной категории носят исключительно информативный характер и никак не влияют на обработку запроса.
2xx	Коды состояния из этой категории возвращаются в случае успешной обработки клиентского запроса.
3xx	Эта категория содержит коды, которые возвращаются, если серверу нужно перенаправить клиента.
4xx	Коды данной категории означают, что на стороне клиента был отправлен некорректный запрос. Например, клиент в запросе указал не поддерживаемый метод или обратился к ресурсу, к которому у него нет доступа.
5xx	Ответ с кодами из этой категории приходит, если на стороне сервера возникла ошибка.

Коды состояния и текст статуса

Полный список кодов состояния доступен в спецификации к протоколу, ниже приведены только самые распространенные коды ответов:

Категория	Описание
200 OK	Возвращается в случае успешной обработки запроса, при этом тело ответа обычно содержит запрошенный ресурс.
302 Found	Перенаправляет клиента на другой URL. Например, данный код может прийти, если клиент успешно прошел процедуру аутентификации и теперь может перейти на страницу своей учетной записи.
400 Bad Request	Данный код можно увидеть, если запрос был сформирован с ошибками. Например, в нем отсутствовали символы завершения строки.
403 Forbidden	Означает, что клиент не обладает достаточными правами доступа к запрошенному ресурсу. Также данный код можно встретить, если сервер обнаружил вредоносные данные, отправленные клиентом в запросе.
404 Not Found	Каждый из нас, так или иначе, сталкивался с этим кодом ошибки. Данный код можно увидеть, если запросить у сервера ресурс, которого не существует на сервере.
500 Internal Error	Данный код возвращается сервером, когда он не может по определенным причинам обработать запрос.

Заголовки ответа

Response Headers, или заголовки ответа, используются для того, чтобы уточнить ответ, и никак не влияют на содержимое тела. Они существуют в том же формате, что и остальные заголовки, а именно «Имя-Значение» с двоеточием (:) в качестве разделителя.

Ниже приведены наиболее часто встречаемые в ответах заголовки:

Категория	Пример	Описание
Server	Server: ngnix	Содержит информацию о сервере, который обработал запрос.
Set-Cookie	Set-Cookie:PHPSSID=bf42938f	Содержит куки, требуемые для идентификации клиента. Браузер парсит куки и сохраняет их в своем хранилище для дальнейших запросов.
WWW-Authenticate	WWW-Authenticate: BASIC realm=«localhost»	Уведомляет клиента о типе аутентификации, который необходим для доступа к запрашиваемому ресурсу.

Тело ответа

Последней частью ответа является его тело. Несмотря на то, что у большинства ответов тело присутствует, оно не является обязательным. Например, у кодов «201 Created» или «204 No Content» тело отсутствует, так как достаточную информацию для ответа на запрос они передают в заголовке.

Пример ответа:

Request URI: http://www.example.com

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Age: 521648
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Fri, 06 Mar 2020 17:36:11 GMT
Etag: "3147526947+gzip"
Expires: Fri, 13 Mar 2020 17:36:11 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (dcb/7EC9)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 648
```

Протокол HTTP и HTTPS

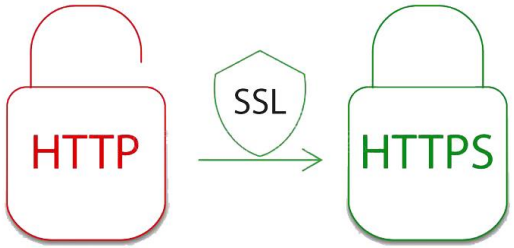
HTTP является расширяемым протоколом, который предоставляет огромное количество возможностей, а также поддерживает передачу всевозможных типов файлов. Однако, вне зависимости от версии, у него есть один существенный недостаток данные передаются в открытом виде. HTTP сам по себе не предоставляет никаких средств шифрования.

HTTPS — это безопасная версия протокола HTTP, которая реализует протокол HTTP с использованием протокола TLS для защиты базового TCP-подключения. За исключением дополнительной конфигурации, необходимой для настройки TLS, использование протокола HTTPS по сути не отличается от протокола HTTP.

Данный протокол по умолчанию использует порт 443.

Протокол активируется для доменов, к которым подключен SSL-сертификат, и вся передача данных идет в зашифрованной форме. Соответственно, ему доверяют больше, т.к. в его основе лежит безопасность работы с информацией.

TLS (англ. transport layer security — протокол защиты транспортного уровня), как и его предшественник SSL (англ. secure sockets layer — слой защищённых сокетов), — криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет. TLS и SSL используют асимметричное шифрование для аутентификации, симметричное шифрование для конфиденциальности и коды аутентичности сообщений для сохранения целостности сообщений.



Как работает HTTPS

1. Пользователь вводит в адресной строке браузера имя сайта.
2. Браузер посылает запрос к серверу на предмет наличия на сайте SSL-сертификата.
3. Сервер отвечает через отправку браузеру копии этого сертификата в комплекте с публичным ключом шифрования.
4. Браузер верифицирует сертификат через запрос в сертификационный центр, его выдавший.
5. После успешного подтверждения браузер создает ключ сеанса, шифрует с помощью полученного публичного ключа и отправляет на сервер.
6. Сервер производит расшифровку сообщения и сохраняет ключ сеанса.
7. Как результат, между сервером и браузером устанавливается безопасное соединение по HTTPS-протоколу.

SSL-сертификат

SSL-сертификаты — это файлы данных, которые электронным способом привязывают ключ шифрования к информации о компании. Если на веб-сервере установлен сертификат, в браузере активируется «замок» и осуществляется безопасное подключение к веб-серверу по протоколу HTTPS.

Обычно SSL используется для безопасных транзакций по кредитным картам, передачи данных и входа с паролем, а в последнее время становится нормой и безопасное подключение к сайтам социальных сетей.

Сертификаты SSL связывают воедино:

- Имя домена, сервера или узла сети.
- Идентификатор организации (например, название компании) и ее местоположение.

Сертификаты SSL используют шифрование с помощью открытого ключа.

Идентификация, аутентификация и авторизация

- Идентификация – процедура, в результате выполнения которой для субъекта выявляется его уникальный признак, однозначно определяющий его 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 прав на выполнение определенных действий.

Схемы http-аутентификации

Распространенные схемы http-аутентификации:

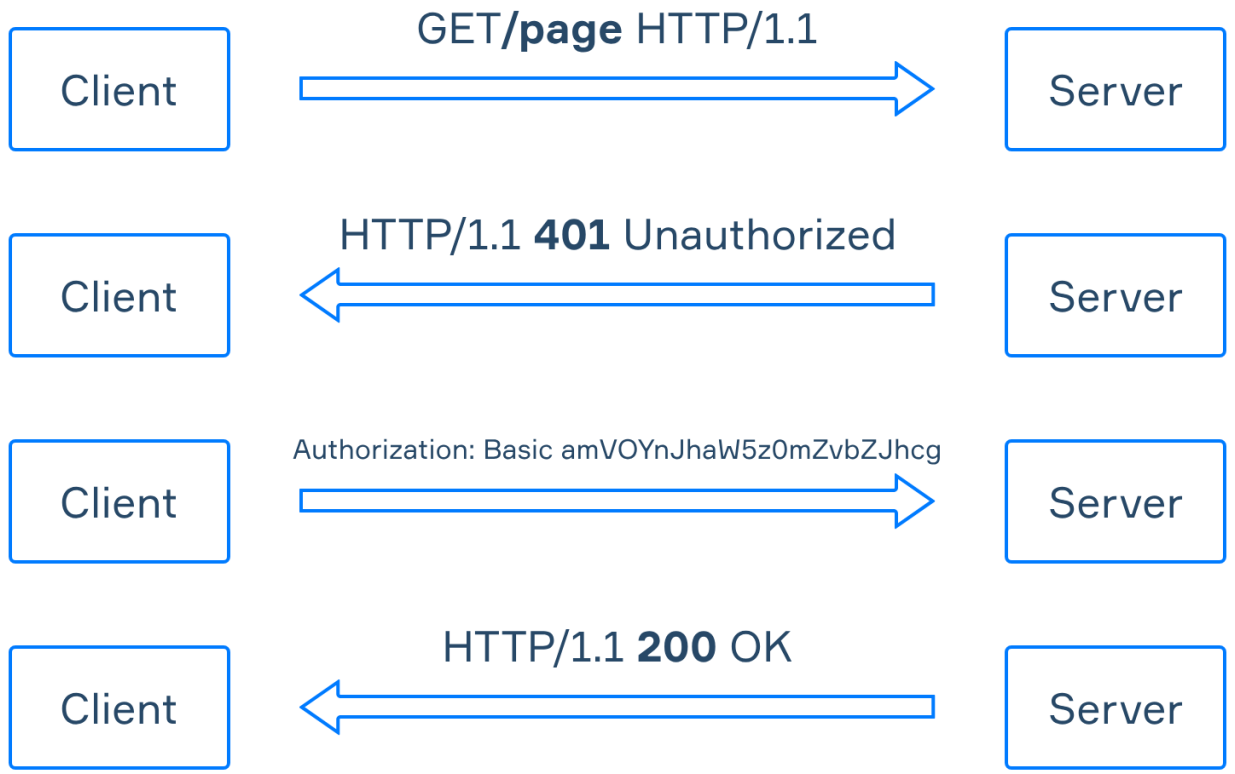
- **Basic** (RFC 7617)
- **Digest** (RFC 7616)
- **Bearer** (RFC 6750)

Полный перечень схем аутентификации можно посмотреть здесь:
https://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml

Базовая (Basic) схема аутентификации

«Базовая» схема HTTP-аутентификации определена в RFC 7617, которая передаёт учётные данные в виде пар пользователь ID/пароль (user ID/password), закодированных с использованием base64.

Поскольку идентификатор (ID) пользователя и пароль передаются по сети в виде открытого текста (он кодируется в base64, однако base64 – это обратимое кодирование), схема базовой аутентификации не является безопасной. При базовой аутентификации следует использовать HTTPS/TLS. Без этих дополнительных улучшений безопасности, базовая аутентификация не должна использоваться для защиты конфиденциальной или ценной информации.



DIGEST AUTHENTICATION

Дайджест-проверка подлинности – это схема запроса-ответа, предназначенная для замены обычной проверки подлинности. Сервер отправляет клиенту строку случайных данных, называемую *nonce*, в качестве запроса. Клиент отвечает с хэшем, который содержит помимо дополнительной информации имя пользователя, пароль и специальное слово. Хэширование данных и сложность этого обмена усложняют кражу и повторное использование учетных данных пользователя при использовании этой схемы проверки подлинности.

Данный метод отправляет по сети хэш-сумму логина, пароля, адреса сервера и случайных данных, и предоставляет больший уровень защиты, чем базовая аутентификация, при которой данные отправляются в открытом виде.

Технически, аутентификация по дайджесту представляет собой применение криптографической хэш-функции MD5 к секрету пользователя с использованием случайных значений для затруднения криптоанализа и предотвращения replay-атак.

Bearer Token аутентификации

Bearer Token или «Токен на предъявителя» – сторона владеющая токеном («предъявитель», bearer), может использовать токен (token) любым способом (как и любая другая сторона, например разные клиенты или приложения). Использование токена не требует от предъявителя доказательства владения. То есть, имея токен, любое приложение может получить доступ к ресурсам.

Токен можно отозвать, для этого на стороне сервиса, выдающего токены должен быть специальный интерфейс, обычно это можно сделать в профиле пользователя и на стороне сервиса или на стороне приложения.

Токен может и должен выдаваться на ограниченный период времени. Так можно уменьшить риск несанкционированного использования.

Токен – это простая строка, текст. Bearer-токен удобно использовать для интеграций, автоматизации и обмена данными между сервисами и приложениями.

OAuth 2

OAuth 2 — это протокол авторизации, предназначенный для организации доступа клиентских приложений к ресурсам, или данным учетных записей, пользователя на другом сервисе.

Мы сталкиваемся с этим протоколом, когда:

- авторизуемся на сторонних площадках через аккаунты соцсетей;
- устанавливаем себе на мобильное устройство приложение, взаимодействующее с нашими данными в облачных сервисах типа Google или Яндекс;
- используем сторонние приложения (боты в Telegram и других мессенджерах) для уведомлений и пр.

Немного практики

Visual Studio Code – редактор для кода

https://code.visualstudio.com/

Одной из особенностей VS Code является то, что он распространяется только с английским языком, а другие языки нужно устанавливать отдельно:

Откройте меню «View – Command Palette» или нажмите F1 или комбинацию клавиш Ctrl-Shift-P и начните вводить фразу «Configure Display». После появления подсказки, кликните на пункт «Configure Display Language».

Среди языков выберите «русский» и нажмите установить. После чего перезапустите редактор.

Live Server для VS Code

Расширение Live Server позволяет запустить свой собственный локальный сервер для разработки и тестирования кода. В нем есть функция «горячей» перезагрузки кода, как для статических страниц, так и для динамических страниц.

- Установите Visual Studio Code (если еще не установлен).
- Зайдите в раздел «Extensions/Расширения» (иконка с блоками на боковой панели) и найдите «Live Server» в поиске.
- Установите расширение, нажав на кнопку «Install/Установить».
- Откройте свой проект в Visual Studio Code и нажмите правой кнопкой мыши на HTML-файле, затем выберите «Open with Live Server» .