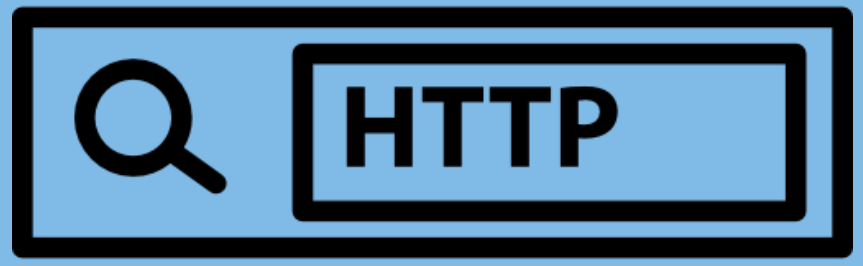
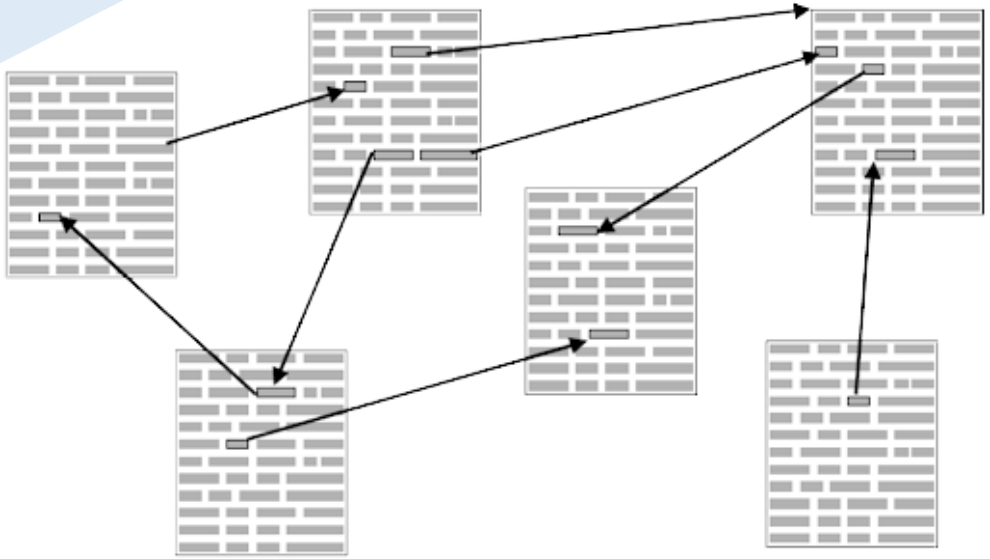
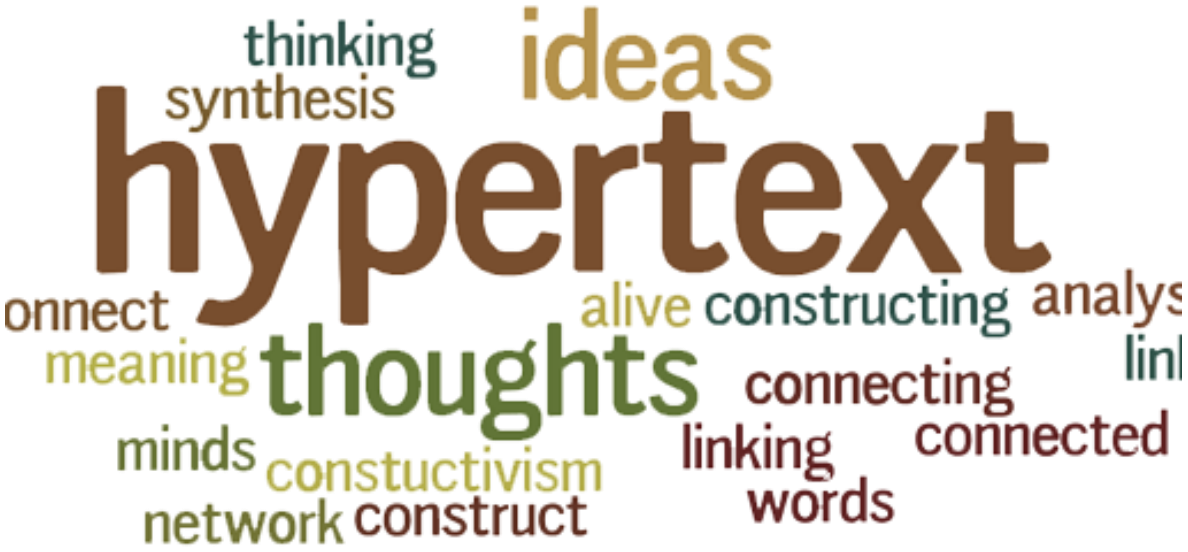
HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP HTTP

HTTP расшифровывается как **Hyper Text Transfer Protocol** - Протокол Передачи Гипертекста.







Гипертекст – это система текстовых страниц, соединенных между собой ссылками. Практически все сайты в интернете представляют собой гипертексты. Этот термин также означает текст, созданный с помощью языка разметки и рассчитанный на использование гиперссылок.

ru.wikipedia.org/wiki/HTTP

Сервисы Gmail YouTube Maps Список для чтения

Вы не представились системе Обсуждение Вклад Создать учётную запись Войти

ВикипедиЯ
Свободная энциклопедия

Статья Обсуждение Читать Текущая версия Править Править код История Искать в Википедии

HTTP

Материал из Википедии — свободной энциклопедии [править | править код]

Стабильная версия была проверена *29 сентября 2021*. Имеются непроверенные изменения в шаблонах или файлах.

HTTP (англ. *HyperText Transfer Protocol* — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

Содержание [скрыть]

HTTP

Название	Hypertext Transfer Protocol
Уровень (по модели OSI)	Прикладной
Семейство	TCP/IP
Создан в	1992
Порт/ID	80/TCP
Спецификация	RFC 1945, RFC 2616, RFC 7231
Основные реализации (клиенты)	Веб-браузеры, например, Internet Explorer, Mozilla Firefox, Opera, Google Chrome, Яндекс.Браузер, Microsoft Edge и др.
Основные реализации (серверы)	Apache, IIS, nginx, Google Web Server и др.

Медиафайлы на Викискладе

Заглавная страница
Рубрикация
Указатель А — Я
Избранные статьи
Случайная страница
Текущие события

Участие
Сообщить об ошибке
Сообщество
Форум
Свежие правки
Новые страницы
Справка
Пожертвовать

Инструменты
Ссылки сюда
Связанные правки
Служебные страницы
Постоянная ссылка
Сведения о странице
Цитировать страницу

Печать/экспорт
Создать книгу
Скачать как PDF
Версия для печати

DevTools is now available in Russian!

Always match Chrome's language | Switch DevTools to Russian | Don't show again

Elements Console Sources Network

```
<!DOCTYPE html>
<html class="client-js ve-available" lang="ru" dir="ltr">
▶<head>…</head>
▼<body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject mw-editable page-HTTP rootpage-HTTP skin-vector action-view skin-vector-legacy">
 == $0
  <div id="mw-page-base" class="noprint"></div>
  <div id="mw-head-base" class="noprint"></div>
▼<div id="content" class="mw-body" role="main">
    <a id="top"></a>
  ▶<div id="siteNotice">…</div>
    <div class="mw-indicators"> </div>
  ▶<h1 id="firstHeading" class="firstHeading">…</h1>
  ▶<div id="bodyContent" class="vector-body">…</div>
  </div>
▼<div id="mw-data-after-content">
    <div class="read-more-container"></div>
  </div>
▶<div id="mw-navigation">…</div>
▶<footer id="footer" class="mw-footer" role="contentinfo">…</footer>
▶<script>…</script>
▶<script type="application/ld+json">…</script>
▶<script>…</script>
```

… s-subject.mw-editable.page-HTTP.rootpage-HTTP.skin-vector.action-view.skin-vector-legacy …

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls

Console Issues

top Filter Default levels 1 Issue: 1

This sidebar will be removed in a future

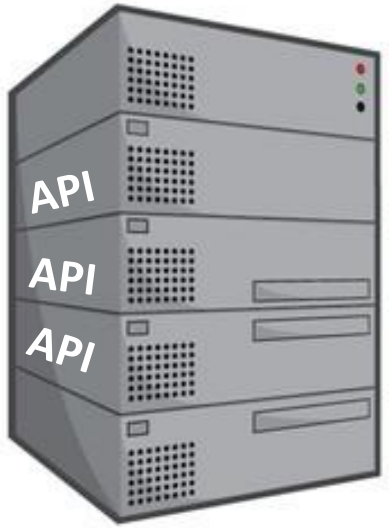
No mess...

Клиент-серверная архитектура

Клиент

Запрос (request)

Ответ (response)
HTML



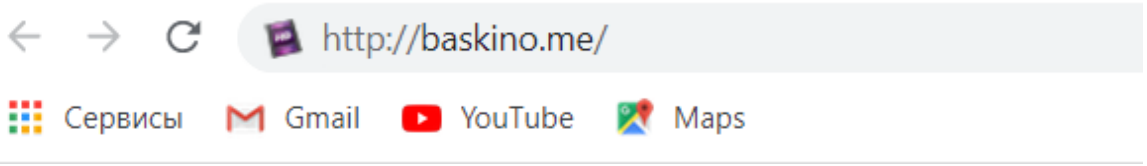
Сервер

SQL

База данных

Как отправить HTTP-запрос?

1



2

тип запроса, одно слово заглавными буквами.

путь к запрашиваемому документу на сервере.

пара разделённых точкой цифр. Например: 1.1.

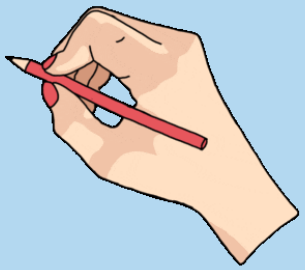
Стартовая строка запроса для HTTP 1.1

Метод URI HTTP/Версия

Headers

Body

Пишем HTTP запрос



GET / HTTP/1.1

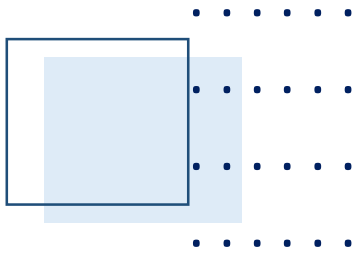
Host: baskino.me

Методы

GET	Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.
HEAD	Запрашивает ресурс так же, как и метод GET, но без тела ответа.
POST	Используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.
PUT	Заменяет все текущие представления ресурса данными запроса.
DELETE	Удаляет указанный ресурс.
OPTIONS	Используется для описания параметров соединения с ресурсом.
PATCH	Используется для частичного изменения ресурса.

CRUD

Create - создание
Read - чтение
Update - обновление
Delete - удаление



GET

Читать - Read

READ

R

GET **Сервер**

Пример запроса

GET /blog/?name1=value1&name2=value2 HTTP/1.1

Host: htmlacademy.ru

Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

HEAD

Читать - Read

READ

R

HEAD **Сервер**

```
HEAD /text.txt HTTP/1.1
Host: example.com
```

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

POST /blog/ HTTP/1.1

Host: htmlacademy.ru

name1=value1&name2=value2

POST

CREATE

C

Create - создание
Update - обновление
Delete - удаление

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PUT

PUT /text.txt HTTP/1.1
Host: example.com

Title=Новый+заголовок+файла
Text=Новый+текст+файла

PUT вносит изменения в уже имеющуюся на сервере информацию.

PUT может сопоставляться как для создания, так и для обновления в зависимости от наличия URI, используемого с PUT.

DELETE

DELETE

D

Удаляет указанный ресурс.


```
DELETE /text.txt HTTP/1.1
Host: example.com
```

Все вышеперечисленные методы можно разделить на три группы:

Безопасные для сервера (GET, HEAD, OPTIONS)

не изменяют данные, их можно выполнять в любой последовательности;

Идемпотентные (GET, HEAD, PUT, DELETE, OPTIONS)

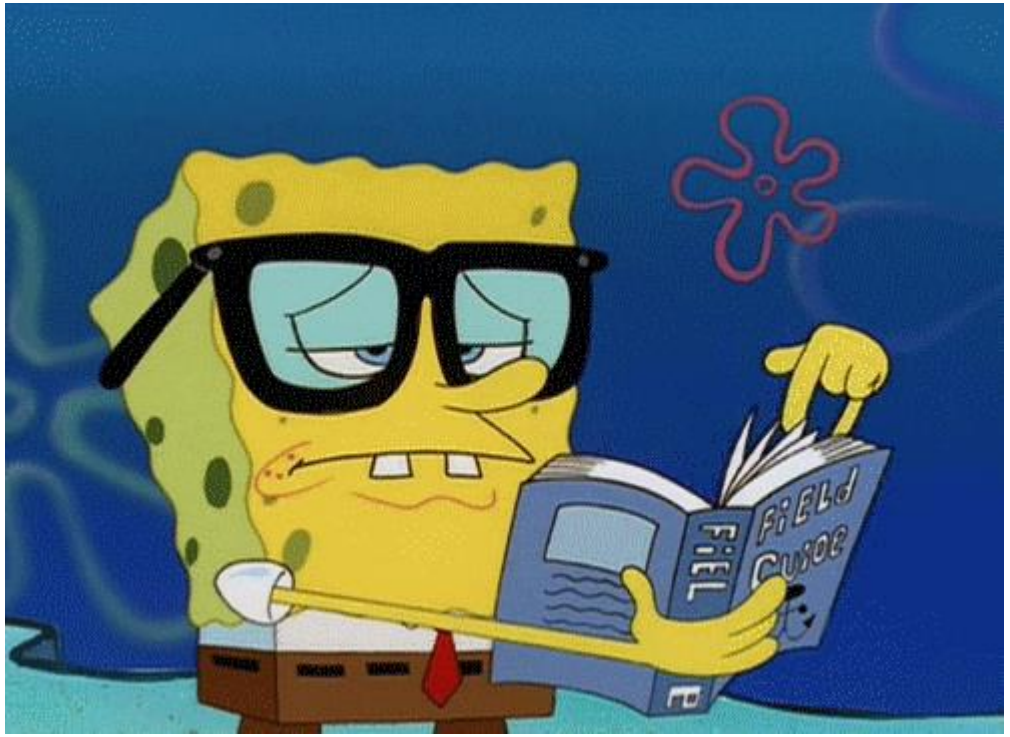
при повторном выполнении результаты будут ожидаемо одинаковыми;

Неидемпотентные (POST, PATCH) —

при повторном выполнении результаты будут разными, если, например, отправить POST-запрос на создание элемента несколько раз подряд, то он может создать несколько элементов с одинаковыми данными.

GET	POST
Кэшируется	Не кэшируется
Тело запроса нет	Тело запроса есть
Остается в истории браузера.	Не остается в истории браузера.
GET запросы могут быть закладками	Запросы POST не могут быть закладками
GET запросы никогда не должны использоваться при работе с конфиденциальными данными	
Да, при отправке данных метод Get добавляет данные в URL-адрес; и длина URL ограничена (максимальная длина URL составляет 2048 символов)	Запросы POST не имеют ограничений по длине данных
GET запросы должны использоваться только для извлечения данных	Обновление, создание, удаление данных
Разрешены только символы ASCII	Никаких ограничений. Двоичные данные также разрешены
Get менее безопасен по сравнению с POST, поскольку отправляемые данные являются частью URL-адреса	POST немного безопаснее, чем Get, поскольку параметры не сохраняются в журнале обозревателя или в журналах веб-сервера
Данные видны всем в URL	Данные не отображаются в URL-адресе
Идемпотентный	Не идемпотентный

Как прочитать ответ?

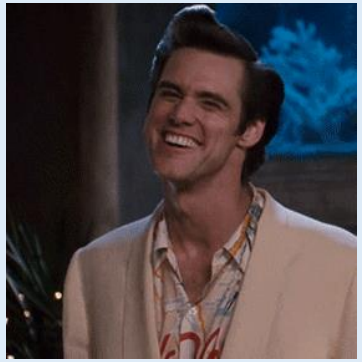


HTTP/Версия	Код состояния	Пояснение
HTTP/1.1	200	OK

```
Access-Control-Allow-Origin: *
Cache-Control: public, max-age=3600
Content-Length: 515523
Content-Type: text/javascript; charset=utf-8
Expires: Wed, 10 Nov 2021 11:06:57 ora solare FLE
Last-Modified: Wed, 01 Jan 2010 01:00:00 GMT
```

Статус коды

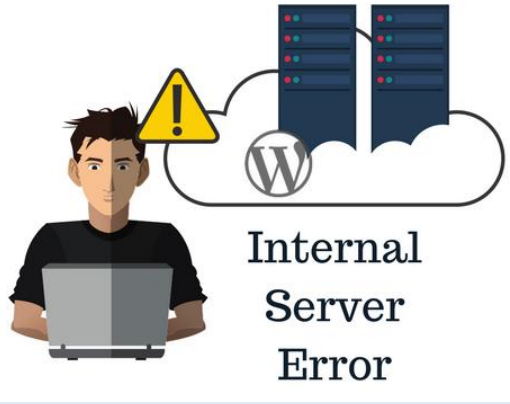
1xx
готовность сервера
информационные



2xx
Успех

3xx
Перенаправление

4xx
Ошибка клиента



5xx
Ошибки сервера

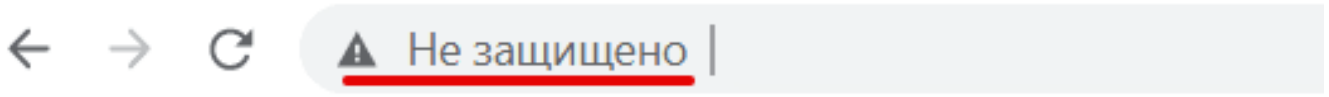
Проблема безопасности

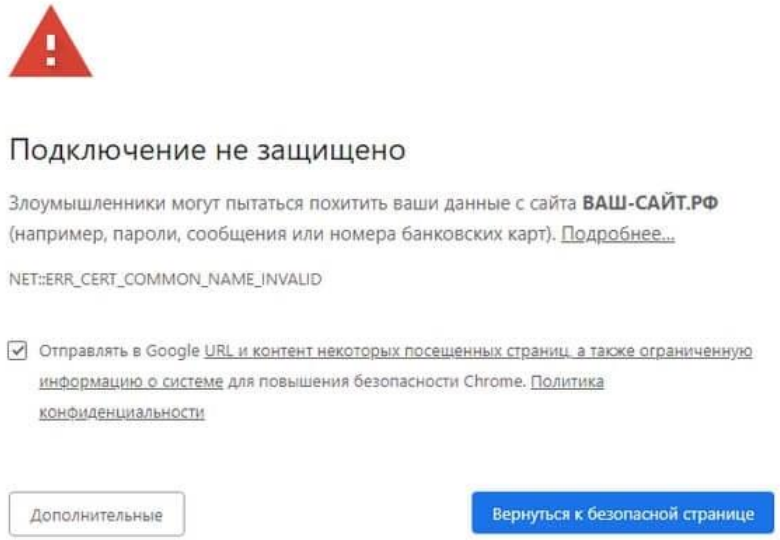
Проблема с HTTP заключается в том, что его изобрели в 1989 году. В то время разработчиков, как и пользователей, не очень беспокоил вопрос интернет-безопасности.

HTTP

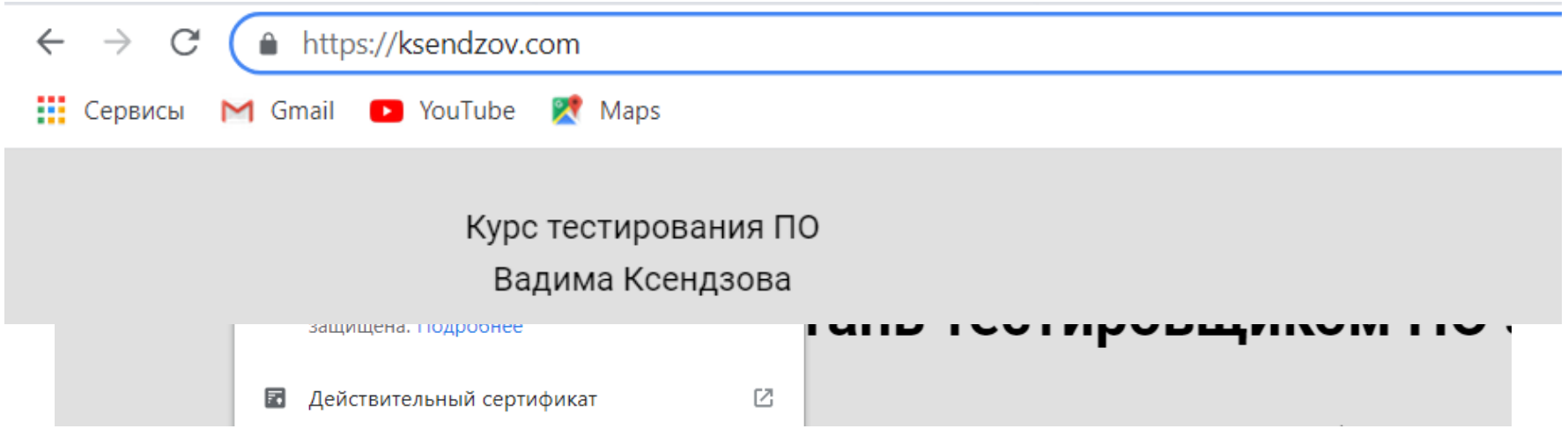
Проблемы с HTTP начались уже в 1990-х, одновременно с глобальным распространением интернета. Появилось много популярных сайтов, на некоторых ввели систему онлайн-платежей.
Тогда и начали активно действовать злоумышленники.

При переходе на сайты с HTTP браузер отображает сообщение о небезопасном подключении.



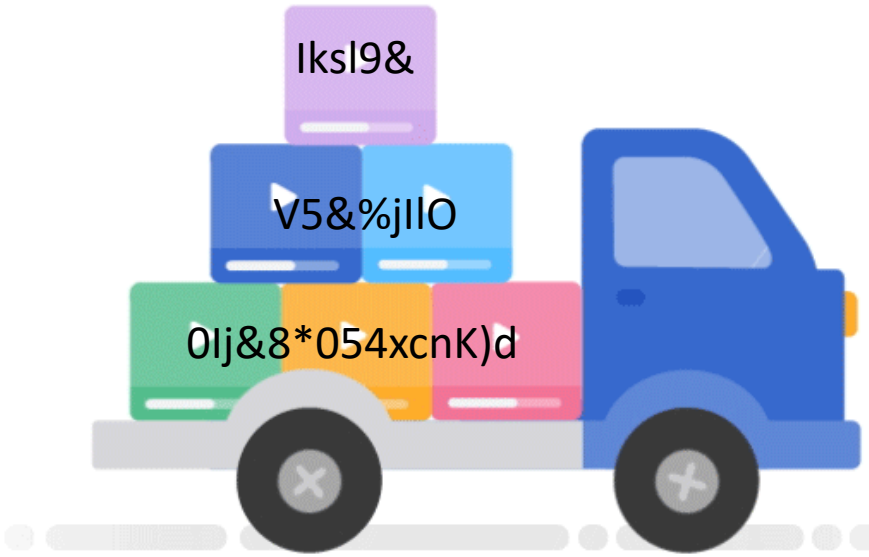


Безопасный протокол гарантирует аутентификацию — попадание пользователей именно на тот ресурс, который необходим, это обеспечивает борьбу с мошенническими вмешательствами.



HTTPS

HyperText Transfer Protocol Secure



В HTTP отсутствует механизм защиты для шифрования данных, в то время как HTTPS для защиты связи между сервером и клиентом использует цифровой сертификат SSL или TLS.

SSL (Secure Sockets Layer)
уровень защищенных сокетов

SSL или слой защищенных сокетов было оригинальным названием протокола, который разработала компания Netscape в середине 90-х. SSL 1.0 никогда не был публично доступным, а в версии 2.0 были серьезные недостатки. Протокол SSL 3.0, выпущенный в 1996, был полностью переделан и задал тон следующей стадии развития.

TLS (Transport Level Security)
безопасность транспортного уровня

Когда следующую версию протокола выпустили в 1999, ее стандартизировала специальная рабочая группа проектирования сети Интернет и дала ей новое название: защита транспортного уровня, или TLS. Как говорится в TLS-документации, «разница между этим протоколом и SSL 3.0 не критичная».

HTTP

Пароль: Настюшка.Опасность99%

HTTPS TLS

Пароль:
Настюшка.Опасность99%

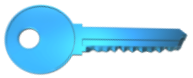
Зашифрованный пароль
V"!o*90fd

Пароль:
Настюшка.Опасность99%

Шифрование
Публичный ключ

Открытый канал связи

Расшифрование
Секретный ключ



RSA Ron Rivest, Adi Shamir, and Leonard Adleman — самый распространенный алгоритм шифрования с использованием асимметричных ключей.

Параметр	HTTP	HTTPS
Название	Hypertext Transfer Protocol	Hypertext Transfer Protocol Secure
Безопасность	Менее безопасен. Данные могут быть доступны для злоумышленников	Он предназначен для предотвращения доступа хакеров к критически важной информации. Защищен атак типа Man-in The-Middle.
Порт	По умолчанию – 80	По умолчанию 443
Начинается на	http://	https://
Область применения	Это хорошо подходит для веб-сайтов общего назначения, таких как блоги.*	Если на сайте нужно вводить конфиденциальную информацию, то данный протокол подходить больше
Защита	Нет защиты передаваемой информации. Любой, кто прослушивает трафик может получить доступ к данным	HTTPS шифрует данные перед передачей их по сети. На стороне получателя, данные расшифровываются.
Протокол	Работает с TCP/IP	Нет специального протокола. Работает поверх HTTP, но использует TLS/SSL шифрование.
Проверка названия домена	Сайтам с HTTP не нужен SSL	Для работы с HTTPS нужен SSL сертификат
Шифрование данных	Не использует шифрование	Данные шифруются
Рейтинг поиска	Не влияет на рейтинг поиска	Помогает увеличивать поисковый рейтинг
Скорость	Быстро**	Относительно медленно
Уязвимость	Уязвима для злоумышленников	Лучше защищен, использует шифрование данных.