**Разница полудуплекс и дуплекс**

*Дуплекс:*

Дуплекс (или полный дуплекс) позволяет устройствам одновременно передавать и принимать данные.

В этом режиме оба конечных устройства могут вести двусторонний разговор, как в обычном телефонном разговоре, где оба собеседника могут говорить и слушать в одно и то же время.

Этот режим обеспечивает максимальную пропускную способность и эффективность связи, но обычно требует более сложной инфраструктуры и оборудования.

*Полудуплекс:*

Полудуплекс (или полудуплексная связь) позволяет устройствам передавать данные только в одном направлении в определенный момент времени.

В этом режиме устройство может либо передавать, либо принимать данные, но не оба действия одновременно. Например, это может быть схоже с использованием рации, где один пользователь говорит, а другой слушает, и затем они меняются ролями.

Полудуплекс обычно менее эффективен с точки зрения пропускной способности, поскольку он не позволяет устройствам одновременно передавать и принимать данные, что может привести к задержкам в обмене информацией.

**Формат передачи данных в разных версиях HTTP**

*HTTP/0.9:*

Формат данных в этой версии очень простой и минималистичный. *Запросы состояли только из строки запроса*, например, "GET /index.html", и *сервер отвечал текстовым HTML-документом*.

Эта версия не поддерживала заголовки и многие другие функции, которые стали стандартом в более поздних версиях HTTP.

*HTTP/1.0:*

Формат данных был расширен, *включая заголовки запроса и ответа*, что позволило передавать дополнительную информацию и метаданные.

В этой версии каждое соединение HTTP закрывалось после того, как сервер передал ответ клиенту.

*HTTP/1.1:*

Включает улучшения, такие как *постоянные соединения, конвейеризацию запросов, поддержку буферизации и множественные запросы на одном соединении*, что снижает задержку при передаче данных.

Формат запроса и ответа включает заголовки, методы запроса (GET, POST, и другие), коды состояния (например, 200 OK, 404 Not Found), и тело сообщения (если необходимо).

*HTTP/2:*

Формат передачи данных оптимизирован для улучшения производительности. Он использует *двоичное кодирование и мультиплексирование (это процесс, при котором несколько потоков данных или сигналов объединяются и передаются через один канал связи или интерфейс)*, что позволяет эффективнее передавать данные и ускоряет загрузку веб-страниц.

В этой версии HTTP заголовки сжимаются, и множественные запросы могут передаваться одним и тем же соединением.

*HTTP/3:*

Основана на протоколе QUIC и предоставляет улучшенную производительность и безопасность передачи данных.

В этой версии HTTP данные передаются по безопасному UDP (User Datagram Protocol) вместо TCP (Transmission Control Protocol), что снижает задержку и ускоряет передачу данных.

**Что такое stateless**

HTTP (Hypertext Transfer Protocol) - это протокол передачи данных в сети, и одним из его ключевых свойств является "stateless" (без состояния). Это означает, что каждый запрос клиента к серверу обрабатывается независимо от предыдущих запросов, и сервер не сохраняет информацию о состоянии клиента между запросами.

В контексте stateless протокола HTTP:

Каждый запрос - это изолированное событие: Когда клиент отправляет HTTP-запрос на сервер, сервер обрабатывает этот запрос без каких-либо знаний о предыдущих запросах от того же клиента. Каждый запрос рассматривается как независимая операция.

Нет сохранения состояния на сервере: HTTP-серверы не сохраняют информацию о состоянии клиента между запросами. Это означает, что сервер не знает, например, какой пользователь выполнил предыдущий запрос или какие данные были отправлены в предыдущих запросах. Каждый запрос должен содержать все необходимые данные для его обработки.

Простота и масштабируемость: Stateless природа HTTP делает его простым и масштабируемым. Сервер не нуждается в хранении состояния для множества клиентов, и это упрощает обработку запросов от большого числа клиентов одновременно.

Однако, для обработки многих типов приложений, которые требуют отслеживания состояния клиента, таких как вход в систему, корзины для покупок и сессии пользователей, часто используются дополнительные механизмы, такие как куки (cookies) и сессии, для создания и сохранения состояния между запросами. Эти механизмы позволяют серверу и клиенту взаимодействовать и сохранять информацию о состоянии дольше, чем один HTTP-запрос, несмотря на изначально stateless природу протокола HTTP.

**Что такое https отличие от http**

HTTP (Hypertext Transfer Protocol) и HTTPS (Hypertext Transfer Protocol Secure) - это две разные версии протокола передачи данных в сети, и их основное отличие заключается в уровне безопасности:

HTTP (Hypertext Transfer Protocol):

HTTP - это стандартный протокол передачи данных в сети, который используется для загрузки и отображения веб-страниц в браузерах.

Однако HTTP не обеспечивает никакой безопасности в процессе передачи данных. Вся информация, передаваемая между клиентом и сервером по протоколу HTTP, передается в открытом виде и может быть прослушана или подвергнута вмешательству третьей стороной.

HTTP прост в использовании и обычно используется для веб-сайтов, где безопасность не является первостепенной задачей.

HTTPS (Hypertext Transfer Protocol Secure):

HTTPS - это расширение HTTP, которое обеспечивает безопасную передачу данных с использованием криптографии.

В протоколе HTTPS вся информация между клиентом и сервером шифруется, что делает сложным перехват и прослушивание передаваемых данных.

Для установления безопасного соединения между клиентом и сервером в протоколе HTTPS используется сертификат SSL/TLS (Secure Sockets Layer/Transport Layer Security).

Веб-сайты, работающие через HTTPS, обычно имеют URL, начинающийся с "https://" вместо обычного "http://". Они также обычно отмечаются зеленым замочком в адресной строке браузера.

Основное отличие между HTTP и HTTPS заключается в том, что HTTPS обеспечивает защиту данных, предотвращая их перехват и модификацию в процессе передачи, что особенно важно при передаче чувствительной информации, такой как пароли, данные кредитных карт и личные данные. В связи с этим, многие веб-сайты, особенно те, которые обрабатывают конфиденциальную информацию, используют HTTPS для обеспечения безопасности своих пользователей.

**Post**

**Методы вкратце**

• GET(идемпотентный) – получение информации: Вы используете этот метод, чтобы запросить информацию с веб-сервера, как если бы вы просили страницу в браузере. Он идемпотентный, что означает, что многократное отправление GET-запроса не изменит состояния сервера

• POST – отправка данных: Этот метод используется для отправки данных на сервер. Например, когда вы заполняете форму на веб-сайте и нажимаете кнопку "Отправить", это создает POST-запрос для передачи данных на сервер.

• PUT(идемпотентный) – обновление данных: PUT-запрос используется для обновления информации на сервере. Вы отправляете данные, которые должны заменить существующие данные на сервере. Это также идемпотентный метод.

• DELETE(идемпотентный) – удаление данных: Как исходит из названия, этот метод используется для удаления информации с сервера. Вы отправляете запрос на удаление определенных данных или ресурса.

• OPTIONS – узнать, что поддерживает сервер: с помощью OPTIONS вы можете узнать, какие методы и возможности поддерживаются на сервере. Это полезно для клиента, чтобы понять, что можно делать на сервере.

• HEAD(идемпотентный) – получение информации без тела: HEAD аналогичен GET, но он не возвращает фактическое содержимое ресурса. Он используется для получения информации о ресурсе, как заголовки ответа, без фактического тела ответа.

• TRACE(идемпотентный) – отслеживание запросов: Этот метод используется для отслеживания пути, который запрос проходит через промежуточные серверы. Это помогает понять, как запрос обрабатывается по пути до сервера.

• CONNECT - устанавливает связь: Этот метод используется для установки двусторонней связи с сервером. Обычно он используется для установления защищенного туннеля, например, для шифрования данных между клиентом и сервером.

**Идемпотентность**

Идемпотентность - это свойство определенных операций или функций, которое означает, что повторное применение операции или функции к тому же входному набору данных или состоянию системы не приведет к изменению результата. В более простых терминах, если операция или функция является идемпотентной, то ее можно выполнять многократно, и результат будет таким же, как при первом выполнении.

Примеры идемпотентных операций в информационных системах и веб-сервисах включают следующее:

HTTP метод GET: Запросы HTTP методом GET идемпотентны. Вы можете многократно выполнять GET-запрос к одному и тому же ресурсу, и он не должен изменить состояние сервера. Это означает, что повторные GET-запросы к одному и тому же URL должны возвращать одинаковый результат.

Удаление (DELETE): Операция удаления в системах управления данными (например, в RESTful API) является идемпотентной. Если вы попытаетесь удалить ресурс несколько раз, результат будет одинаковым: ресурс будет удален.

**Как передать параметры в url**

**как посмотреть заголовки в браузере**

**какой движок у Node.js**

Node.js - это среда выполнения для JavaScript, а не движок. Однако внутри Node.js используется движок JavaScript для выполнения кода JavaScript. Движок JavaScript, который Node.js использует по умолчанию, называется V8.

V8 - это открытый и быстрый движок JavaScript, разработанный Google. Он компилирует JavaScript в машинный код и выполняет его с высокой производительностью. V8 широко используется не только в Node.js, но и в браузере Google Chrome, что делает его одним из самых быстрых движков JavaScript.

Node.js обеспечивает среду выполнения для JavaScript на стороне сервера, позволяя разработчикам создавать серверные приложения и микросервисы с использованием языка JavaScript. Node.js также предоставляет множество встроенных модулей и инструментов для работы с сетью, файлами, вводом/выводом и другими задачами, что делает его популярным выбором для создания высокопроизводительных и масштабируемых серверных приложений.

**Require**