**1. Интернет**

**Интернет** — это глобальная сеть, состоящая из взаимосвязанных компьютерных систем, использующих стандартизированные протоколы связи для обмена данными. Он обеспечивает доступ к различным ресурсам и услугам, таким как веб-сайты, электронная почта и файлообмен.

**2. Компоненты интернета**

Основные компоненты интернета включают:

* **Устройства**: компьютеры, серверы, маршрутизаторы, коммутаторы и другие сетевые устройства.
* **Протоколы**: Набор правил для передачи данных (например, TCP/IP, HTTP, FTP).
* **Сети**: Локальные и глобальные сети, соединяющие устройства.
* **Сервисы**: Веб-сервисы, электронная почта, VoIP и т.д.
* **Клиенты и серверы**: Устройства, инициирующие запросы (клиенты) и обрабатывающие эти запросы (серверы).

**3. Клиент-серверная архитектура**

**Клиент-серверная архитектура** — это модель взаимодействия, где один узел (клиент) запрашивает ресурсы или услуги у другого узла (сервера).

**Типы взаимодействия**:

* **Запрос-ответ**: Клиент отправляет запрос, сервер обрабатывает его и отправляет ответ (например, HTTP).
* **Поток данных**: Сервер может отправлять данные клиенту в реальном времени (например, WebSocket).
* **Отложенные запросы**: Клиент может отправлять запросы без ожидания немедленного ответа (например, REST API).

**4. Веб-приложение**

**Веб-приложение** — это приложение, доступное через веб-браузер, которое использует веб-технологии для выполнения задач. Оно может включать интерактивные элементы и динамическое взаимодействие с пользователем через серверные скрипты.

**5. Свойства HTTP**

* **Протокол прикладного уровня**: HTTP работает на верхнем уровне модели OSI.
* **Запрос-ответный**: Клиент отправляет запрос, сервер отвечает.
* **Текстовый протокол**: Сообщения в HTTP могут быть прочитаны человеком.
* **Статус кодов**: HTTP использует коды состояния для обозначения успеха или неудачи запросов.
* **Расширяемость**: Можно добавлять заголовки и методы.

**6. Stateless протокол**

**Stateless протокол** — это протокол, который не сохраняет информацию о состоянии между запросами. Каждый запрос от клиента к серверу рассматривается как независимый. Примером является HTTP, где сервер не удерживает информацию о предыдущих взаимодействиях.

**7. HTTPS**

**HTTPS** (HTTP Secure) — это защищенная версия HTTP, использующая шифрование (TLS/SSL) для обеспечения безопасности передачи данных между клиентом и сервером. Это защищает от перехвата и подмены данных.

**8. Структура запроса**

Структура HTTP-запроса включает:

* **Метод**: Тип запроса (GET, POST и т.д.).
* **URI**: Уникальный ресурс, к которому обращается клиент.
* **Версия протокола**: Версия HTTP (например, HTTP/1.1).
* **Заголовки**: Дополнительные параметры запроса (например, User-Agent, Content-Type).
* **Тело запроса**: Данные, отправляемые серверу (например, при POST-запросе).

**9. Группы заголовков**

* **Общие заголовки**: Применяются к обеим сторонам (например, Cache-Control).
* **Заголовки запроса**: Применяются только к запросу (например, Accept).
* **Заголовки ответа**: Применяются только к ответу (например, Location).
* **Заголовки сущности**: Содержат информацию о содержимом (например, Content-Type).

**10. Методы запроса**

* **GET**: Запрашивает данные.
* **POST**: Отправляет данные на сервер.
* **PUT**: Обновляет данные на сервере.
* **DELETE**: Удаляет данные.
* **HEAD**: Запрашивает заголовки, без тела ответа.
* **OPTIONS**: Запрашивает поддерживаемые методы для ресурса.
* **PATCH**: Отправляет только те данные, которые нужно изменить.
* **CONNECT:** Устанавливает туннельный соединение с сервером, часто используется с протоколом HTTPS через прокси-сервер.
* **TRACE:** Позволяет клиенту увидеть, что сервер получает и как обрабатывает запрос.

**11. Структура URI**

URI (Uniform Resource Identifier) состоит из URL и URN.

Состоит из:

* **Схема**: Протокол (например, http, https).
* **Хост**: Доменное имя или IP-адрес.
* **Порт**: Номер порта (по умолчанию 80 для HTTP и 443 для HTTPS).
* **Путь**: Путь к ресурсу на сервере.
* **Запрос**: Параметры запроса, начинающиеся с ?.

**12. Идемпотентность**

**Идемпотентность** — это свойство методов HTTP, при котором повторное выполнение одного и того же запроса не изменяет состояние сервера после первого выполнения. Методы GET и DELETE являются идемпотентными, тогда как POST — нет.

**13. Структура ответа**

Структура HTTP-ответа включает:

* **Версия протокола**: Версия HTTP, используемая сервером.
* **Код состояния**: Код, указывающий на результат обработки запроса.
* **Фраза состояния**: Описание кода состояния.
* **Заголовки**: Дополнительная информация о ответе.
* **Тело ответа**: Данные, отправленные клиенту.

**14. Коды состояния**

Коды состояния делятся на категории:

* **1xx**: Информационные.
* **2xx**: Успешные (например, 200 OK).
* **3xx**: Перенаправления (например, 301 Moved Permanently).
* **4xx**: Ошибки клиента (например, 404 Not Found).
* **5xx**: Ошибки сервера (например, 500 Internal Server Error).

**15. Что такое сессия**

**Сессия** — это период взаимодействия между клиентом и сервером, обычно с сохранением состояния. Сессии позволяют отслеживать действия пользователя на сайте, например, хранить информацию о входе в систему.

**16. Что такое куки**

**Куки** — это небольшие данные, хранящиеся на стороне клиента, которые используются для сохранения информации о пользователе между сессиями. Куки могут содержать данные о состоянии, предпочтениях, аутентификации и т.д.

**Способы передачи параметров от клиента на сервер**

* **URL-параметры**: Используются в строке запроса URL (например, ?key=value).
* **Тело запроса**: Используется в POST-запросах для передачи данных.
* **Заголовки**: Дополнительные данные могут передаваться в заголовках HTTP-запроса.

**Способы сохранить состояние в HTTP**

* **Куки**: Хранят данные на стороне клиента.
* **Сессии**: Хранят данные на сервере и связывают их с уникальным идентификатором, который хранится в куках.
* **URL-параметры**: Могут использоваться для передачи состояния между страницами.

TLS (Transport Layer Security) и его предшественник SSL (Secure Sockets Layer) используют процесс рукопожатия для установления защищенного соединения между клиентом и сервером. Вот основные этапы алгоритма TLS/SSL рукопожатия:

**Алгоритм рукопожатия TLS/SSL**

1. **Инициализация клиента**:
   * Клиент отправляет сообщение ClientHello, содержащее:
     + Версию протокола TLS/SSL, которую он поддерживает.
     + Список поддерживаемых шифров (cipher suites).
     + Список поддерживаемых методов сжатия.
     + Случайное число (random nonce).
2. **Ответ сервера**:
   * Сервер отвечает сообщением ServerHello, которое включает:
     + Выбранную версию протокола.
     + Выбранный шифр.
     + Список поддерживаемых методов сжатия.
     + Случайное число от сервера.
3. **Аутентификация и обмен ключами**:
   * Сервер отправляет свой сертификат (Certificate), который содержит открытый ключ.
   * Дополнительно сервер может отправить:
     + Сообщение ServerKeyExchange (для некоторых шифров).
     + Сообщение CertificateRequest (если сервер требует аутентификации клиента).
   * Сервер отправляет сообщение ServerHelloDone, указывающее, что его часть рукопожатия завершена.
4. **Ответ клиента**:
   * Клиент проверяет сертификат сервера.
   * Если требуется аутентификация, клиент отправляет свой сертификат (Certificate).
   * Клиент отправляет сообщение ClientKeyExchange, содержащее информацию для создания общего секретного ключа.
   * Клиент создает и отправляет сообщение ChangeCipherSpec, указывающее, что дальнейшие сообщения будут зашифрованы.
   * Клиент отправляет Finished, чтобы подтвердить завершение рукопожатия.
5. **Завершение на стороне сервера**:
   * Сервер также отправляет сообщение ChangeCipherSpec, указывающее, что он будет использовать зашифрованный режим.
   * Сервер отправляет Finished, чтобы подтвердить завершение рукопожатия.

**Итог**

После завершения всех шагов рукопожатия клиент и сервер используют общий секретный ключ для шифрования и дешифрования последующих данных, что обеспечивает безопасность соединения.