9. Дайте определение понятию «Интернет».

Глобальная система взаимосвязанных компьютерных сетей, которая позволяет компьютерам и устройств по всему миру обмениваться данными и ресурсами.

10. Дайте определение понятию «Служба Интернет».

Онлайн-сервис, предоставляемый через Интернет, который может предоставлять различные функциональные возможности, такие как электронная почта, поисковые системы, социальные сети, облачное хранилище и другие. Эти службы обычно доступны через веб-браузер или специализированные приложения

11. Дайте определение понятию «Узел сети Интернет».

Конкретный компьютер или устройство, подключенное к Интернету и способное взаимодействовать с другими узлами для обмена данными. Узел может быть как сервером, предоставляющим ресурсы, так и клиентом, запрашивающим ресурсы.

12. Дайте определение понятию «клиент-серверное приложение».

Архитектурный шаблон в разработке программного обеспечения, где одни компоненты (клиенты) запрашивают и получают услуги или ресурсы от других компонентов (серверов). Клиенты и серверы взаимодействуют через сеть, часто посредством сетевых протоколов.

13. Дайте определение понятию «сетевой протокол».

Набор правил и соглашений, которые определяют формат и последовательность обмена данными между устройствами в сети. Протоколы определяют, как информация упаковывается, передается и распаковывается, чтобы обеспечить надежную и эффективную связь.

14. Перечислите основные свойства протокола HTTP.

* версии HTTP/1.1 – действующий (текстовый), HTTP/2 – черновой (не распространен, бинарный), HTTP/3 (раньше HTTP-over-QUIC, стадия черновика);
* два типа абонентов: клиент и сервер;
* два типа сообщений: request и response;
* от клиента к серверу – request;
* от сервера к клиенту – response;
* на один request всегда один response, иначе ошибка;
* одному response всегда один request, иначе ошибка;
* stateless;
* TCP-порты: 80, 443;
* для адресации используется URI или URL;
* поддерживается W3C, описан в нескольких RFC.

15. Перечислите состав информации, пересылаемой в HTTP-запросе.

* Метод запроса (GET, POST, PUT, DELETE)
* URI (Uniform Resource Identifier) - идентификатор запрашиваемого ресурса.
* Заголовки запроса - метаданные о запросе.
* Тело запроса (не всегда присутствует) - данные, отправляемые на сервер.

16. Перечислите состав информации, пересылаемой в HTTP-ответе.

* Код состояния - трехзначный код, который указывает на результат выполнения запроса (например, 200 - OK, 404 - Not Found).
* Заголовки ответа - метаданные о ответе.
* Тело ответа (не всегда присутствует) - данные, возвращаемые сервером.

17. Перечислите группы заголовков HTTP.

* **Общие заголовки:** применяются к обоим запросам и ответам.
* **Заголовки запроса:** предоставляют информацию о запросе.
* **Заголовки ответа:** предоставляют информацию о ответе.
* **Заголовки сущности:** предоставляют информацию о сущности, передаваемой или возвращаемой.

18. Дайте определение понятию «web-приложение».

* программное приложение, которое доступно и используется через веб-браузер посредством сети Интернет или локальной сети. Веб-приложения основаны на веб-технологиях, таких как HTML, CSS и JavaScript, и выполняются на сервере, взаимодействуя с клиентскими браузерами через протокол HTTP (или его защищенный вариант, HTTPS).

19. Дайте определение понятиям «frontend» и «backend».

* **Frontend** - это часть веб-приложения, с которой взаимодействует пользователь. Это включает в себя пользовательский интерфейс, который отображается в браузере, и клиентскую логику, которая выполняется на стороне клиента (например, JavaScript).
* **Backend** - это серверная часть веб-приложения, которая обрабатывает запросы от клиентов, выполняет бизнес-логику, взаимодействует с базами данных и отправляет данные клиентам. Backend обычно работает на сервере.

20. Дайте определение понятию «кроссплатформенное приложение».

Приложение, которое может работать на разных операционных системах или платформах без значительных изменений. Кроссплатформенные приложения обычно разрабатываются с использованием специальных технологий и инструментов, которые обеспечивают совместимость с разными средами выполнения.

21. Изобразите и поясните общую схему web-приложения.

22. Назовите основные технологии разработки серверных кроссплатформенных приложений.

- **Node.js:** Среда выполнения JavaScript, которая позволяет разрабатывать серверные приложения на JavaScript.

**- ASP.NET Core** Фреймворк для разработки серверных приложений на языках C# и F#.

- **Java**: Язык программирования и экосистема Java EE для создания серверных приложений.

- **Python:** Язык программирования Python с библиотеками, такими как Django и Flask, для создания серверных приложений.

- **Ruby on Rails:** Фреймворк Ruby для разработки серверных приложений.

- **PHP**: Язык программирования PHP с фреймворками, такими как Laravel и Symfony.

- **GO**: Язык программирования Go, известный как Golang, для создания высокопроизводительных серверных приложений.

23. Поясните понятие «асинхронная операция».

Операция, которая выполняется независимо от остальной части программы, и результат ее выполнения может быть получен позже, без блокирования выполнения других операций. Асинхронные операции часто используются в веб-программировании для выполнения задач, которые могут занять некоторое время, такие как загрузка данных из базы данных или запрос к стороннему API, без блокирования интерфейса пользователя.

24. Поясните основное назначение NODE.JS.

Среда выполнения JavaScript, построенная на движке Chrome V8, которая позволяет выполнять JavaScript на сервере. Основным назначением Node.js является создание масштабируемых и высокопроизводительных сетевых приложений, таких как веб-серверы. Node.js поддерживает асинхронное программирование, что делает его идеальным для обработки множества одновременных запросов.

25. Перечислите основные свойства NODE.JS.

- **Асинхронность:** Node.js поддерживает асинхронное выполнение операций, что позволяет обрабатывать множество запросов одновременно без блокировки.

- **Высокая производительность:** Node.js обладает высокой производительностью благодаря использованию многопоточного и неблокирующего ввода-вывода.

- **Модульность:** Node.js поддерживает модульную структуру, что упрощает организацию кода и повторное использование.

- **Обширная библиотека:** Node.js имеет обширную библиотеку модулей ипакетов, доступных через систему управления пакетами npm (Node Package Manager).

- **Скорость разработки**: Node.js позволяет быстро разрабатывать приложения благодаря использованию JavaScript как языка программирования и доступу к множеству готовых модулей.

- **Сообщество**: Node.js имеет активное сообщество разработчиков, что обеспечивает поддержку и развитие экосистемы.

- **Поддержка WebSocket**: Node.js отлично подходит для создания приложений, использующих WebSocket для двусторонней связи между клиентом и сервером.

Эти свойства делают Node.js популярным выбором для разработки серверных приложений, веб-серверов, API и других сетевых приложений, где требуется высокая производительность и масштабируемость.