Лабораторная работа 01

ПвИ

**Задание 01**

1. Установите Node.js.
2. Проверьте работоспособность Node.js.

**Задание 02**

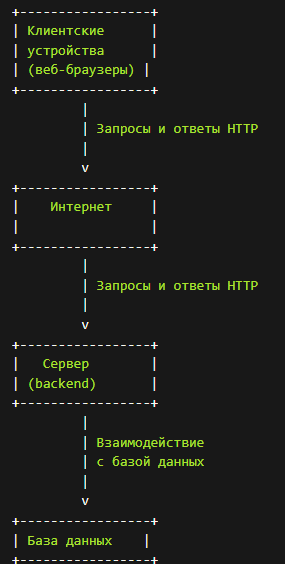
1. Разработайте серверное приложение **01-01**, использующее модуль http и возвращающее ответ с разметкой ***<h1>Hello World</h1>*.**
2. Проверьте работоспособность приложения с помощью браузера.
3. Проверьте работоспособность приложения с помощью приложения ***POSTMAN***.

**Задание 03**

1. Разработайте серверное приложение **01-02**, на основе разработанного в задании 2, которое в ответе сервера пересылает html-страницу с содержимым запроса (метод, uri, версия протокола, заголовки, тело).
2. Выполните GET-запрос к серверу с помощью браузера и убедитесь в работоспособности сервера.
3. Выполните POST-запрос (содержащий данные в теле сообщения) к серверу с помощью POSTMAN и убедитесь в работоспособности сервера.

**Задание 04.** Ответьте на следующие вопросы.

1. Дайте определение понятию «Интернет».
2. Дайте определение понятию «Служба Интернет».
3. Дайте определение понятию «Узел сети Интернет».
4. Дайте определение понятию «клиент-серверное приложение».
5. Дайте определение понятию «сетевой протокол».
6. Перечислите основные свойства протокола HTTP.
7. Перечислите состав информации, пересылаемой в HTTP-запросе.
8. Перечислите состав информации, пересылаемой в HTTP-ответе.
9. Перечислите группы заголовков HTTP.
10. Дайте определение понятию «web-приложение».
11. Дайте определение понятиям «frontend» и «backend».
12. Дайте определение понятию «кроссплатформенное приложение».
13. Изобразите и поясните общую схему web-приложения.
14. Назовите основные технологии разработки серверных кроссплатформенных приложений.
15. Поясните понятие «асинхронная операция».
16. Поясните основное назначение NODE.JS.
17. Перечислите основные свойства NODE.JS.
18. Интернет - это глобальная сеть компьютеров, которая соединяет миллионы устройств по всему миру. Он обеспечивает обмен информацией и коммуникацию между пользователями, позволяя им получать доступ к различным ресурсам, таким как веб-сайты, электронная почта, файлы и т.д.
19. Служба Интернет - это сервис или набор сервисов, предоставляемых провайдерами Интернета или другими организациями для пользователей Интернета. Они могут включать в себя доступ к сети Интернет, электронную почту, хостинг веб-сайтов, облачное хранилище данных и другие функции, которые позволяют пользователям использовать Интернет для своих нужд.
20. Узел сети Интернет - это устройство или компьютер, подключенное к Интернету и способное передавать и получать данные. Узлы сети Интернет могут быть серверами, компьютерами, маршрутизаторами, переключателями, мобильными устройствами и другими устройствами, которые играют роль в передаче данных через Интернет.
21. Клиент-серверное приложение - это модель распределенного приложения, в которой клиентские устройства (клиенты) обращаются к серверу для получения данных или выполнения определенных операций. Клиенты отправляют запросы серверу, а сервер обрабатывает эти запросы и возвращает ответы с необходимыми данными или результатами операций.
22. Сетевой протокол - это набор правил и форматов, которые определяют, как устройства в сети обмениваются данными и устанавливают соединение. Протоколы определяют структуру данных, способы кодирования и декодирования информации, а также методы проверки доставки и обработки ошибок в сети.
23. Основные свойства протокола HTTP (Hypertext Transfer Protocol):
24. HTTP является протоколом прикладного уровня, используемым для передачи гипертекстовых документов в виде веб-страниц.
25. Он основан на клиент-серверной модели, где клиент отправляет запросы серверу, а сервер возвращает ответы.
26. HTTP является безсостоятельным протоколом, то есть он не сохраняет состояние между запросами от клиента.
27. Протокол HTTP работает поверх протокола TCP/IP и использует порт 80 для обмена данными.
28. Информация, пересылаемая в HTTP-запросе, включает:
29. Метод запроса (например, GET, POST, PUT, DELETE) для указания типа операции, которую нужно выполнить на сервере.
30. URI (Uniform Resource Identifier) или URL (Uniform Resource Locator) для указания ресурса, с которым нужно взаимодействовать.
31. Заголовки запроса, содержащие дополнительные метаданные, такие как тип контента, язык, кэширование и другие параметры.
32. Опциональное тело запроса, которое может содержать данные, передаваемые на сервер.
33. Информация, пересылаемая в HTTP-ответе, включает:
34. Статусный код, который указывает на результат выполнения запроса (например, 200 OK, 404 Not Found, 500Internal Server Error).
35. Заголовки ответа, содержащие метаданные о ответе, такие как тип контента, длина контента, дата и другие параметры.
36. Опциональное тело ответа, которое содержит данные, возвращаемые клиенту.
37. Группы заголовков HTTP включают:
38. Заголовки общего назначения, которые применяются к обоим запросам и ответам, например, заголовок "Connection" для указания типа соединения.
39. Заголовки запроса, которые содержат информацию о запросе, такую как "User-Agent" (информация о клиенте) и "Host" (адрес сервера).
40. Заголовки ответа, которые содержат информацию о ответе, такую как "Content-Type" (тип контента) и "Content-Length" (длина контента).
41. Заголовки сущности, которые содержат информацию о самом теле запроса или ответа, например, "Content-Encoding" (кодировка контента) и "Content-Language" (язык контента).
42. Web-приложение - это программное приложение, которое выполняется на удаленном сервере и доступно для пользователей через Интернет с помощью веб-браузера. Оно обычно состоит из клиентской части (frontend), которая работает на стороне пользователя, и серверной части (backend), которая обрабатывает запросы клиента и взаимодействует с базами данных или другими внешними ресурсами.
43. Frontend - это часть веб-приложения, которая работает на стороне клиента (веб-браузера) и отвечает за отображение пользовательского интерфейса и взаимодействие с пользователем. Он использует языки программирования, такие как HTML, CSS и JavaScript, для создания веб-страниц, стилей и интерактивности.
44. Backend - это часть веб-приложения, которая работает на сервере и отвечает за обработку запросов от клиента, взаимодействие с базами данных и другими внешними сервисами, и отправку данных обратно клиенту. Он использует различные языки программирования, такие как PHP, Python, Java или Node.js, для создания логики приложения и обработки данных.
45. Кроссплатформенное приложение - это приложение, которое может работать на разных операционных системах или платформах без изменений или с небольшими модификациями. Кроссплатформенные приложения обычно разрабатываются с использованием фреймворков или инструментов, которые обеспечивают абстракцию от конкретных особенностей платформы, позволяя разработчикам создавать единый код, который можно запускать на разных устройствах.



В данной схеме клиентские устройства, такие как компьютеры или мобильные устройства, запускают веб-браузеры, которые отправляют HTTP-запросы на сервер. Сервер (backend) обрабатывает эти запросы, взаимодействует с базой данных (если это необходимо) и возвращает HTTP-ответы обратно клиенту. Клиентские устройства получают ответы и отображают соответствующую информацию пользователю.

Основные технологии разработки серверных кроссплатформенных приложений включают:

1. Node.js: Среда выполнения JavaScript, которая позволяет разрабатывать серверные приложения на JavaScript.
2. ASP.NET Core: Фреймворк для разработки серверных приложений на языке C#.
3. Django: Фреймворк на языке Python для разработки веб-приложений.
4. Ruby on Rails: Фреймворк на языке Ruby для разработки веб-приложений.
5. Laravel: Фреймворк на языке PHP для разработки веб-приложений.
6. Express.js: Минималистичный фреймворк на Node.js для разработки веб-приложений.

Асинхронная операция - это операция, которая выполняется независимо от основного потока выполнения программы. Вместо блокировки основного потока, пока операция не завершится, асинхронная операция запускается и выполняется параллельно с другими операциями. Основной поток может продолжать свою работу, а по завершении асинхронной операции будет выполнена соответствующая обработка результата. Примеры асинхронных операций включают запросы к базе данных, загрузку файлов из сети или ожидание ответа от удаленного сервера.

Node.js - это среда выполнения JavaScript, которая позволяет запускать JavaScript-код на сервере. Основное назначение Node.js заключается в создании масштабируемых и эффективных серверных приложений. Он обеспечивает асинхронную и событийно-ориентированную модель ввода/вывода, что позволяет обрабатывать большое количество одновременных подключений без блокировки основного потока выполнения. Node.js также имеет широкий выбор модулей и инструментов для разработки серверных приложений, что делает его популярным выбором для создания веб-серверов, API и других серверных приложений.

Основные свойства Node.js:

1. Асинхронность: Node.js использует асинхронную модель ввода/вывода, что позволяет обрабатывать множество одновременных операций без блокировки основного потока выполнения.
2. СобытиОсновные свойства Node.js:
3. Асинхронность: Node.js использует асинхронную модель ввода/вывода, что позволяет обрабатывать множество одновременных операций без блокировки основного потока выполнения. Это делает Node.js эффективным для обработки большого количества одновременных запросов.
4. Событийно-ориентированная архитектура: Node.js основан на событийно-ориентированной архитектуре, где определенные события могут вызывать обработчики. Это позволяет разработчикам создавать эффективные и отзывчивые приложения, которые реагируют на события в реальном времени.
5. Единый язык программирования: Node.js использует язык программирования JavaScript как для клиентской, так и для серверной разработки. Это позволяет разработчикам использовать один язык программирования на всем стеке технологий, упрощая разработку и обмен кодом между клиентом и сервером.
6. Модульность: Node.js поддерживает модульную структуру, позволяя разработчикам создавать и использовать модули для повторного использования кода. Существует широкий выбор модулей, доступных через пакетный менеджер npm, что упрощает разработку и интеграцию сторонних библиотек.
7. Высокая производительность: Благодаря своей асинхронной и событийно-ориентированной модели, Node.js обладает высокой производительностью и масштабируемостью. Он может эффективно обрабатывать большое количество одновременных запросов и является популярным выбором для создания высоконагруженных серверных приложений.
8. Кросс-платформенность: Node.js доступен на различных операционных системах, включая Windows, macOS и Linux, что обеспечивает кросс-платформенную совместимость и позволяет разработчикам создавать приложения, которые могут работать на различных системах.