Лабораторная работа 09

ПСКП

**Задание 01**

1. Разработайте приложение **09-01**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. WS-сервер предназначен для приема по ws-каналу файлов.
3. Принятый по ws-каналу файл записывается в директорию **upload**.
4. Разработайте приложение **09-01a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера.

**Задание 02**

1. Разработайте приложение **09-02**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. WS-сервер предназначен для отправки по ws-каналу файлов из директории **download**.
3. Разработайте приложение **09-02a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера.

**Задание 03**

1. Разработайте приложение **09-03**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Сервер каждые 15 секунд всем подключившимся клиентам высылает сообщение следующего формата:

**09-03-server: *n***, где ***n*** - последовательный номер отправляемого сервером сообщения.

1. С помощью ***ping/pong***-механизма сервер проверяет работоспособность соединений, каждые 5 секунд, при этом сервер выводит на консоль количество работоспособных соединений.
2. Разработайте приложение **09-03a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера.Продемонстрируйте работу сервера с несколькими экземплярами **09-03a**.

**Задание 04**

1. Разработайте приложение **09-04**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Сервер принимает сообщение вида:

{client:***x***, timestamp:***t***}, где ***x***-имя клиента, а ***t***–штамп времени.

Сообщение передается клиентом в json-формате.

1. Сервер отправляет в ответ клиенту сообщение вида:

{server: **n** client:***x***, timestamp:***t***}, где ***n*** –номер сообщения, ***x***-имя клиента, а ***t***–штамп времени.

Сообщение передается сервером в json-формате.

1. Разработайте приложение **09-04a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение принимает параметр командной строки, значение которого используется в качестве значения ***x*,** в сообщении для сервера.
2. Продемонстрируйте взаимодействие сервера с несколькими клиентами (клиенты должны иметь разные значения параметра).

**Задание 05**

1. Разработайте приложение **09-05**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Сервер обеспечивает следующий RPC-интерфейс:

|  |  |  |
| --- | --- | --- |
| RPC  метод | public  protected | Описание RPC-метода |
| square | public | если принимает один параметр ***r***, то возвращается площадь круга радиуса r;  если принимает два параметрa ***a*** и ***b***, то возвращается площадь прямоугольника с длинами сторон ***a*** и ***b***; |
| sum | public | принимает переменное количество числовых параметров, возвращает сумму значений всех параметров; |
| mul | public | принимает переменное количество числовых параметров, возвращает произведение значений всех параметров; |
| fib | protected | принимает один числовой параметр ***n***, возвращает массив, содержащий ***n***  элементов последовательности Фибоначчи; |
| fact | protected | принимает один числовой параметр ***n***, возвращает факториал числа ***n***; |

1. Разработайте приложение **09-05a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение осуществляет следующие RPC-вызовы:

***square(3), square(5,4),***

***sum(2), sum(2,4,6,8,10),***

***mul(3), mul(3,5,7,9,11,13),***

***fib(1), fib(2), fib(7),***

***fact(0), fact(5), fact(10)***

1. Результаты вычислений отобразите в консоли приложения.
2. Разработайте приложение **09-05b,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение осуществляет параллельные (async/parallel) RPC-вызовы из п.20. Результаты вычислений отобразите в консоли приложения.
3. Разработайте приложение **09-05с,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение вычисляет с помощью RPC-вызовов следующее выражение:

***sum(square(3), square(5,4), mul(3,5,7,9,11,13))***

***+fib(7)***

***\*mul(2,4,6)***

Результаты вычислений отобразите в консоли приложения.

**Задание 06**

1. Разработайте приложение **09-06**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Приложение может генерировать три события: ***A, B, C.***
3. Генерация событий осуществляется при получении соответствующего сообщения через стандартный поток ввода (через консоль). При вводе символа ***A*** сервер генерирует событие ***A***;при вводе символа ***B*** сервер генерирует событие ***B***; при вводе символа ***C*** сервер генерирует событие ***C***.
4. Разработайте приложение **09-06a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение подписывается на событие ***A*** и сообщает о наступлении этого события выводом на консоль.
5. Разработайте приложение **09-06b,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение подписывается на событие ***B*** и сообщает о наступлении этого события выводом на консоль.
6. Разработайте приложение **09-06c,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение подписывается на событие ***C*** и сообщает о наступлении этого события выводом на консоль.
7. Продемонстрируйте совместную работу всех четырех приложений.

**Задание 07**

1. Разработайте приложение **09-07**, представляющее собой WebSocket(WS)-север, прослушивающий порт **4000**.
2. Приложение может принимать три типа уведомлений: ***A, B, C.*** При получении уведомления, сервер выводит соответствующее сообщение на консоль.
3. Разработайте приложение **09-07a,** представляющее собой WS-клиент, демонстрирующий работоспособность сервера. Приложение шлет уведомления серверу при получении соответствующего сообщения через стандартный поток ввода (консоль).

**Задание 08.** Ответьте на следующие вопросы.

1. Поясните понятие «TCP-порт».
2. Поясните понятие «сетевой сокет».
3. Поясните понятие «WebSocket».
4. Поясните процедуру установки соединения между WS-сервером и WS-клиентом.
5. Поясните понятие «широковещательное сообщение».
6. Поясните принцип организации потокового ввода/вывода через WS-соединение.
7. Поясните принцип действия и назначение механизма «ping/pong».
8. Поясните аббревиатуру «RPC».
9. Поясните принцип работы RPC-механизма, предоставляемого пакетом rpc-websockets.
10. Поясните принцип действия и назначение механизма «subscriber/publisher».
11. Поясните принцип действия и назначение механизма уведомлений.
12. Поясните понятие «TCP-порт».

TCP-порт — это числовой идентификатор, используемый для адресации процесса в компьютерной сети. В контексте Node.js и сетевого программирования, TCP-порты используются для определения конечной точки соединения между двумя узлами в сети. Когда приложение запускает сервер на определенном порту, оно слушает входящие соединения на этом порту, чтобы устанавливать обмен данными с другими приложениями.

1. Поясните понятие «сетевой сокет».

Сетевой сокет представляет собой программный интерфейс для обеспечения обмена данными между процессами, работающими на разных узлах сети. В контексте Node.js, сетевые сокеты могут использоваться для установки соединения между сервером и клиентом через протоколы, такие как TCP или UDP.

1. Поясните понятие «WebSocket».

WebSocket — это протокол обмена сообщениями между клиентом и сервером через одно постоянное соединение в режиме реального времени. Этот протокол обеспечивает более эффективную и меньшую задержку связь по сравнению с традиционными методами HTTP. В Node.js существуют библиотеки, такие как ws, которые позволяют легко реализовывать WebSocket-серверы и клиенты.

1. Поясните процедуру установки соединения между WS-сервером и WS-клиентом.

Процедура установки соединения между WS-сервером и WS-клиентом:

WS-сервер слушает определенный порт.

WS-клиент устанавливает соединение с сервером, отправляя запрос с заголовком Upgrade: websocket.

Сервер подтверждает запрос, и соединение становится WebSocket-соединением.

Теперь сервер и клиент могут обмениваться данными в режиме реального времени через это соединение.

1. Поясните понятие «широковещательное сообщение».

Широковещательное сообщение — это сообщение, отправляемое одним отправителем и получаемое всеми узлами в сети. В контексте Node.js и сетевого программирования, широковещательные сообщения могут использоваться для передачи данных от сервера ко всем подключенным клиентам.

1. Поясните принцип организации потокового ввода/вывода через WS-соединение.

Принцип организации потокового ввода/вывода через WS-соединение:

Потоковый ввод/вывод через WebSocket-соединение в Node.js может быть реализован с использованием объектов потока (Stream). Данные, поступающие через соединение, могут быть обработаны как поток, что обеспечивает эффективность и возможность обработки больших объемов данных по мере их поступления.

1. Поясните принцип действия и назначение механизма «ping/pong».

Механизм "ping/pong" в WebSocket используется для поддержания активности соединения. Сервер или клиент могут отправлять "ping" (запрос) и ожидать "pong" (ответа). Если одна из сторон не получает "ping" в течение определенного времени, соединение может считаться разорванным.

1. Поясните аббревиатуру «RPC».

RPC — это механизм взаимодействия между удаленными процессами или программами, который позволяет вызывать процедуры или функции на удаленном сервере так, как если бы они были вызваны локально.

1. Поясните принцип работы RPC-механизма, предоставляемого пакетом rpc-websockets.

rpc-websockets — это библиотека для реализации RPC в приложениях на основе WebSocket. С ее помощью можно определять удаленные процедуры, вызывать их на сервере через WebSocket-соединение и получать результаты

1. Поясните принцип действия и назначение механизма «subscriber/publisher».

Механизм "subscriber/publisher" используется для реализации шаблона "издатель-подписчик". Клиенты могут подписываться на определенные события (издатели), и при возникновении этих событий сервер отправляет уведомления подписчикам.

1. Поясните принцип действия и назначение механизма уведомлений.

Механизм уведомлений позволяет серверу отправлять сообщения клиентам о каких-то событиях, даже если клиент не явно запрашивает эти данные. Это особенно полезно в реальном времени, когда сервер хочет уведомить клиента о изменениях или событиях, произошедших на сервере.