**Экзаменационные вопросы по дисциплине**

**«Программирование серверных кроссплатформенных приложений»**

**для студентов 3-го курса специальности ПОИТ**

1. Протокол HTTP, основные свойства HTTP, структура запроса и ответа. Понятие web-приложения, структура и принципы работы web-приложения. Понятие асинхронности.
2. Web-сервер. Ресурсы, потребляемые web-сервером. Блокирующие и неблокирующие операции ввода/вывода. Решение проблемы блокирующего ввода/вывода. Понятия конкурентность и параллельность. Закон Амдала.
3. Протокол WebSockets, основные свойства, процедура установки соединения. WebSockets API.
4. Платформа Node.js, версии, назначение, основные свойства, структура, принципы работы, основные встроенные модули и их назначение, применение внешних модулей (пакетов). Web-приложение «Hello World». Пример.
5. Глобальные объекты Node.js (global, process) и их применение. Системные (стандартные потоки) Node.js (stdin, stdout, stderr) и их применение. Модуль console: функции log, error, dir, time, timeEnd, trace. Примеры.
6. Асинхронное программирование. Функция обратного вызова. Проблема "Callback hell" и способы решения. Примеры.
7. Асинхронное программирование. Механизм Promises. Механизм async/await. Примеры.
8. Класс EventEmitter, назначение, применение. Пример.
9. Функции setTimeout, setInterval, nextTick, ref, unref, назначение, применение. Примеры.
10. Модули и пакеты Node.js, функция require, кэширование модуля, область видимости в пакете, экспорт объектов, функций, конструкторов. Применение require для работы с json-файлами. Параметризируемый модуль. Пример.
11. Пакетный менеджер NPM, глобальное хранилище, просмотр установленных пакетов, скачивание пакетов, назначение файла package.json, локальные хранилища пакетов, удаление пакетов, публикация пакета. Примеры.
12. Разработка простейшего HTTP-сервера в Node.js. Извлечение данных из HTTP-запроса, формирование данных HTTP-ответа. Пример. Тестирование с помощью POSTMAN.
13. Разработка простейшего HTTP-сервера в Node.js. Извлечение данных из HTTP-запроса, формирование данных HTTP-ответа. Пример. Тестирование с помощью браузера AJAX (XMLHTTPRequest/Fetch).
14. Разработка HTTP-сервера в Node.js. Обработка GET, POST, PUT и DELETE-запросов. Генерация ответа с кодом 405. Пример. Тестирование с помощью POSTMAN.
15. Разработка HTTP-сервера в Node.js. Обработка URI HTTP-запроса, маршрутизация запросов, генерация ответа с кодом 404. Пример. Тестирование с помощью POSTMAN.
16. Разработка HTTP-сервера в Node.js. Обработка запросов к статическим ресурсам: html, css, js, png, msword. Пример. Тестирование с помощью браузера.
17. Разработка HTTP-сервера в Node.js. Обработка query-параметров GET-запроса. Пример. Тестирование с помощью браузера.
18. Разработка HTTP-сервера в Node.js. Обработка path-параметров GET-запроса. Пример. Тестирование с помощью браузера.
19. Разработка HTTP-сервера в Node.js. Обработка параметров POST-запроса. Пример. Тестирование с помощью браузера (<form>) и POSTMAN.
20. Разработка HTTP-сервера в Node.js. Обработка json-сообщения в POST-запросе. Пример. Тестирование с помощью POSTMAN.
21. Разработка HTTP-сервера в Node.js. Обработка xml-сообщения в POST-запросе. Пример. Тестирование с помощью POSTMAN.
22. Разработка HTTP-сервера в Node.js. Пересылка файла в POST-запросе (upload). Пример. Тестирование с помощью браузера.
23. Разработка HTTP-сервера в Node.js. Пересылка файла в ответе (download). Пример. Тестирование с помощью браузера.
24. Разработка HTTP-клиента в Node.js. Отправка GET запроса с query-параметрами. Пример. Тестирование с помощью с Node.js-сервера.
25. Разработка HTTP-клиента в Node.js. Отправка POST-запроса с параметрами в теле. Пример. Тестирование с помощью с Node.js-сервера.
26. Разработка HTTP-клиента в Node.js. Отправка POST-запроса с json-сообщением. Пример. Тестирование с помощью с Node.js-сервера.
27. Разработка HTTP-клиента в Node.js. Обработка json-ответа. Пример. Тестирование с помощью с Node.js-сервера.
28. Разработка HTTP-клиента в Node.js. Обработка xml-ответа. Пример. Тестирование с помощью с Node.js-сервера.
29. Разработка HTTP-клиента в Node.js. Пересылка файла на сервер в POST-запросе (upload). Пример. Тестирование с помощью с Node.js-сервера.
30. Разработка HTTP-клиента в Node.js. Обработка ответа с файлом (download). Пример. Тестирование с помощью с Node.js-сервера.
31. Разработка Websockets-приложения: Node.js-сервер, браузер-клиент. Пример.
32. Разработка широковещательного Websockets-приложения: Node.js-сервер, Node.js-клиент. Пример.
33. Разработка Websockets-приложения: Node.js-сервер с применением потока, Node.js-клиент. Пример.
34. Разработка Websockets-приложения: ping/pong-сообщения, Node.js-сервер, Node.js-клиент. Пример.
35. Разработка Websockets-приложения: обработка json-сообщений, Node.js-сервер, Node.js-клиент. Пример.
36. Разработка Websockets-приложения: отправка клиентом файла (upload), Node.js-сервер, Node.js-клиент. Пример.
37. Разработка Websockets-приложения: отправка сервером файла (download), Node.js-сервер, Node.js-клиент. Пример.
38. Разработка RPC-Websockets-сервера. Пример. Тестирование: Node.js-клиент.
39. Разработка RPC-Websockets-сервера: обработка уведомлений. Пример. Тестирование: Node.js-клиент.
40. Работа с файловой системой в Node.js: создание, копирование, проверка существования файла, запись, запись в конец, чтение, синхронные асинхронные операции. Пример.
41. Работа с файловой системой в Node.js: создание, удаление, переименование, запись, запись в конец, чтение, синхронные асинхронные операции. Пример.
42. Работа с файловой системой в Node.js: создание, слежение за файлом, запись, запись в конец, чтение, синхронные асинхронные операции. Пример.
43. Работа с файловой системой в Node.js: запись в файл потока октетов, чтение из файла потока октетов. Пример.
44. Работа с файловой системой в Node.js: запись в файл массива 32-битовых целочисленных данных, чтение из файла массива 32-битовых целочисленных данных. Пример.
45. Применение потокового чтение (Readable) и записи (Writable) файлов в Node.js. Пример.
46. Применение функции pipe для обработки данных (файла) запроса и записи в файл файловой системы. Пример.
47. Применение функции pipe для обработки данных (файла) файловой системы и записи в http-ответ. Пример.
48. Разработка клиент-серверного TCP-приложения: обмен текстовыми сообщениями. Пример.
49. Разработка клиент-серверного TCP-приложения: пересылка массива целочисленных данных. Пример.
50. Разработка клиент-серверного TCP-приложения: пересылка файла от клиента серверу. Пример.
51. Разработка клиент-серверного TCP-приложения: пересылка файла от сервера клиенту. Пример.
52. Разработка клиент-серверного TCP-приложения прослушивающего два порта, обмен текстовыми сообщениями. Пример.
53. Разработка клиент-серверного UDP-приложения: обмен текстовыми сообщениями. Пример.
54. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического SELECT-запроса.
55. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического INSERT-запроса. Пример.
56. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического UPDATE-запроса. Пример.
57. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического DELETE-запроса. Пример.
58. Разработка приложения, выполняющего запрос к SQL-базе данных: вызов удаленной процедуры. Пример.

1. Разработка приложения, выполняющего graphql-запрос к SQL-базе данных: query-запрос. Пример.
2. Разработка приложения, выполняющего graphql-запрос к SQL-базе данных: mutation-запрос. Пример.

асс. каф. ИСиТ М.В. Дубовик

**В билете 3 вопроса: 1 и 2 вопросы из списка (1-60), 3-й вопрос – демонстрация одной из лабораторных работ (1-16).**

**На экзамене студент обязан предоставить все выполненные (1-16) лабораторные работы. Студент, который не предоставит полный список выполненных лабораторных работ автоматически получает неудовлетворительную оценку.**

**В качестве дополнительного вопроса может быть затронута тема, рассмотренная на лекции, но не вошедшая в список экзаменационных вопросов (MongoDB, Mongoose, REST, SemVer и др.).**