«УТВЕРЖДАЮ»
Зав. кафедрой ИСиТ_____ В.В. Смелов

Экзаменационные вопросы дисциплины
«Программирование серверных кроссплатформенных приложений»
для студентов 3-го курса специальности ПОИТ

В билете 3 вопроса: 1 и 2 вопросы из списка (1-57), 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.

1. Протокол HTTP, основные свойства HTTP, структура запроса и ответа. Протокол HTTPS. Понятие web-приложения, структура и принципы работы web-приложения. Понятие асинхронности.

2. Протокол WebSockets, основные свойства, процедура установки соединения. WebSockets API.

3. 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.

4. 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.

5. Класс EventEmitter, назначение, применение. Пример.

6. Функции setTimeout, setInterval, nextTick, ref, unref, назначение, применение. Примеры.

7. 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.

8. Пакетный менеджер NPM, глобальное хранилище, просмотр установленных пакетов, сканирование пакетов

просмотр установленных пакетов, скачивание пакетов, назначение файла package.json, локальные хранилища пакетов, удаление пакетов, публикация пакета. Примеры.

9. Разработка простейшего HTTP-сервера в Node.js. Извлечение данных из HTTP-запроса, формирование данных HTTP-ответа. Пример. Тестирование с помощью POSTMAN.

10. Разработка простейшего HTTP-сервера в Node.js. Извлечение данных из HTTP-запроса, формирование данных HTTP-ответа. Пример. Тестирование с помощью браузера AJAX (XMLHTTPRequest/Fetch).

11. Разработка HTTP-сервера в Node.js. Обработка GET, POST, PUT и DELETE-запросов. Генерация ответа с кодом 404. Пример. Тестирование с помощью POSTMAN.

12. Разработка HTTP-сервера в Node.js. Обработка URI HTPP-запроса, маршрутизация запросов, генерация ответа с кодом 404. Пример. Тестирование с помощью POSTMAN.

13. Разработка HTTP-сервера в Node.js. Обработка запросов к статическим ресурсам: html, css, js, png, msword. Пример. Тестирование с помощью браузера.

14. Разработка HTTP-сервера в Node.js. Обработка query-параметров GET-запроса. Пример. Тестирование с помощью браузера.

15. Разработка HTTP-сервера в Node.js. Обработка uri-параметров GET-запроса. Пример. Тестирование с помощью браузера.

16. Разработка HTTP-сервера в Node.js. Обработка параметров POST-запроса. Пример. Тестирование с помощью браузера (<form>) и POSTMAN.

17. Разработка HTTP-сервера в Node.js. Обработка json-сообщения в POST-запросе. Пример. Тестирование с помощью POSTMAN.

18. Разработка HTTP-сервера в Node.js. Обработка xml-сообщения в POST-запросе. Пример. Тестирование с помощью POSTMAN.

19. Разработка HTTP-сервера в Node.js. Пересылка файла в POST-запросе (upload). Пример. Тестирование с помощью браузера.

20. Разработка HTTP-сервера в Node.js. Пересылка файла в ответе (download). Пример. Тестирование с помощью

в ответе (download). Пример. Тестирование с помощью браузера.

21. Разработка HTTP-клиента в Node.js. Оправка GET запроса с query-параметрами. Пример. Тестирование с помощью с Node.js-сервера.
22. Разработка HTTP-клиента в Node.js. Оправка POST-запроса с параметрами в теле. Пример. Тестирование с помощью с Node.js-сервера.
23. Разработка HTTP-клиента в Node.js. Оправка POST-запроса с json-сообщением. Пример. Тестирование с помощью с Node.js-сервера.
24. Разработка HTTP-клиента в Node.js. Обработка json-ответа. Пример. Тестирование с помощью с Node.js-сервера.
25. Разработка HTTP-клиента в Node.js. Обработка xml-ответа. Пример. Тестирование с помощью с Node.js-сервера.
26. Разработка HTTP-клиента в Node.js. Пересылка файла на сервер в POST-запросе (upload). Пример. Тестирование с помощью с Node.js-сервера.
27. Разработка HTTP-клиента в Node.js. Обработка ответа с файлом (download). Пример. Тестирование с помощью с Node.js-сервера.
28. Разработка Websockets-приложения: Node.js-сервер, браузер-клиент. Пример.
29. Разработка широковещательного Websockets-приложения: Node.js-сервер, Node.js-клиент. Пример.
30. Разработка Websockets-приложения: Node.js-сервер с применением потока, Node.js-клиент. Пример.
31. Разработка Websockets-приложения: ping/pong-сообщения, Node.js-сервер, Node.js-клиент. Пример.
32. Разработка Websockets-приложения: обработка json-сообщений, Node.js-сервер, Node.js-клиент. Пример.
33. Разработка Websockets-приложения: отправка клиентом файла (upload), Node.js-сервер, Node.js-клиент. Пример.
34. Разработка Websockets-приложения: отправка сервером файла (download), Node.js-сервер, Node.js-клиент. Пример.
35. Разработка RPC-Websockets-сервера. Пример.

Тестирование: Node.js-клиент.

36. Разработка RPC-Websockets-сервера: обработка уведомлений. Пример. Тестирование: Node.js-клиент.

37. Работа с файловой системой в Node.js: создание, копирование, проверка существования файла, запись, запись в конец, чтение, синхронные асинхронные операции. Пример.

38. Работа с файловой системой в Node.js: создание, удаление, переименование, запись, запись в конец, чтение, синхронные асинхронные операции. Пример.

39. Работа с файловой системой в Node.js: создание, слежение за файлом, запись, запись в конец, чтение, синхронные асинхронные операции. Пример.

40. Работа с файловой системой в Node.js: запись в файл потока октетов, чтение из файла потока октетов. Пример.

41. Работа с файловой системой в Node.js: запись в файл массива 32-битовых целочисленных данных, чтение из файла массива 32-битовых целочисленных данных. Пример.

42. Применение потокового чтение (Readable) и записи (Writable) файлов в Node.js. Пример.

43. Применение функции pipe для обработки данных (файла) запроса и записи в файл файловой системы. Пример.

44. Применение функции pipe для обработки данных (файла) файловой системы и записи в http-ответ. Пример.

45. Разработка клиент-серверного TCP-приложения: обмен текстовыми сообщениями. Пример.

46. Разработка клиент-серверного TCP-приложения: пересылка массива целочисленных данных. Пример.

47. Разработка клиент-серверного TCP-приложения: пересылка файла от клиента серверу. Пример.

48. Разработка клиент-серверного TCP-приложения: пересылка файла от сервера клиенту. Пример.

49. Разработка клиент-серверного TCP-приложения прослушивающего два порта, обмен текстовыми сообщениями. Пример.

50. Разработка клиент-серверного UDP-приложения: обмен

50. Разработка клиент-серверного UDP-приложения: обмен текстовыми сообщениями. Пример.

51. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического SELECT-запроса.

52. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического INSERT-запроса. Пример.

53. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического UPDATE-запроса. Пример.

54. Разработка приложения, выполняющего запрос к SQL-базе данных: выполнение динамического DELETE-запроса. Пример.

55. Разработка приложения, выполняющего запрос к SQL-базе данных: вызов удаленной процедуры. Пример.

56. Разработка приложения, выполняющего graphql-запрос к SQL-базе данных: query-запрос. Пример.

57. Разработка приложения, выполняющего graphql-запрос к SQL-базе данных: mutation-запрос. Пример.

Доцент каф. ИСиТ
В.В. Смелов