# Лабораторная работа №1

1. **Дайте определение понятию «Интернет».**

всемирная сеть, построенная на протоколах tcp\ip

1. **Дайте определение понятию «Служба Интернет».**

системы, предоставляющие некоторые услуги (dns, www, ftp, telnet, smtp, pop3, imap, ...)

1. **Дайте определение понятию «Узел сети Интернет».**

устройство, соединённое с другими устройствами как часть компьютерной сети. Узлами могут быть компьютеры, телефоны, а также маршрутизаторы, коммутаторы, etc.

1. **Дайте определение понятию «клиент-серверное приложение».**

Которое состоит из клиента (фронт) и сервера (бек) которые общаются реквестами и респонсами

1. **Дайте определение понятию «сетевой протокол».**

сетевой протокол - http

1. **Перечислите основные свойства протокола HTTP.**

* прикладной уровень, юзается в вебе для вз-ия клиента и сервера
* 1.1 основная версия, но много сайтов перешли на http 2 (отличие: протокол бинарный, http 3.0 разрабатывается, работает поверх протокола похожего на udp)
* подразумевает клиентов и серверов (клиент - реквест - сервер), (сервер-респонс-клиент)
* не сохраняет состояния, каждый раз не сохраняет инфу о клиенте (фикс с пом. сессий)
* http - 80, https - 4043, rfc стандарт

1. **Перечислите состав информации, пересылаемой в HTTP-запросе.**

инфа в хттп-запросе - метод (get, post, put, delete, copy, options, ..), uri, версия, заголовки (key\value с доп инфой о соединении, кеше, куки,..), параметры, тело

заголовки - пустая строка - тело

1. **Перечислите состав информации, пересылаемой в HTTP-ответе.**

инфа в хттп-ответе - версия, код состояния (100 - 599), пояснение к коду состояния, заголовки - пустая строка - тело

1. **Дайте определение понятию «web-приложение».**

веб-приложение - клиент-серверное приложение, у которого клиент и сервер вз-ют по протоколу http (не всегда,но в основном). состоит из фронта (клиента) и бека (сервер)

1. **Дайте определение понятиям «frontend» и «backend».**

* фронт - клиент, дизигн
* бек - сервер, бизнес-логика

1. **Дайте определение понятию «кроссплатформенное приложение».**

кроссплатформерное приложение - способно работать на более чем одной аппаратной платформе. 2 способа: на уровне компилятора (c, c++) или на уровне среды (фреймворка) - jvm, .net clr, node.js - после компиляции получаем байт код (il), который запускается на виртуальной машине

1. **Изобразите и поясните общую схему web-приложения.**

То же самое что и в клиент-серверном приложении

1. **Назовите основные технологии разработки серверных** **кроссплатформенных приложений.**

php, jvm, asp.net core, python (django), ruby (rails), js (node.js)

1. **Поясните понятие «асинхронная операция».**

асинхронная операция - поток не блочится, задачи отдаются на выполнение, 2 этапа: запрос и получение результата

1. **Перечислите основные свойства сервера NODE.JS.**

ориентирован на i/o операции (input/output) - не связанные с вычислением (для них cpu).

# Лабораторная работа №2

**1. Перечислите основные свойства сервера NODE.JS**

главное - это асинхронность: код будет выполняться не последовательно, а сразу, как только будет готов. также все операции выполняются в одном потоке.

сервер ноды событийноориентирован: почти на каждую операцию имеются коллбэки, а также можно легко создавать свои события. события строятся по схеме паттерна reactor, который применяет eventloop

**2. Что такое npm?**

встроенный в ноду менеджер пакетов, который управляет внешними зависимостями. с его помощью можно установить новую зависимость

**3. Поясните назначение HTTP-заголовка Content-Type.**

означает, какой будет тип получаемого в ходе респонса ресурса: text/html, text/plain, image/jpg, ...

**4. Поясните назначение функции require.**

функция реквайр подключает внешние и встроенные модули. все основные функции ноды расположены в отдельных модулях (файлах js), которые просто импортируются в код нашей программы. аналог #include (c++)\using(C#)\import (react)

**5. Поясните понятие «Модуль Node.js».**

модуль - некоторая часть встроенного функционала ноды. все стандартные функции разбиты по пакетам (модулям), которые надо подключать в начале каждого файла

**6. Поясните понятие «Node.js built-in modules» («Node.js Core modules»).**

встроенные модули - стандартные модули ноды, которые идут с ней в комплекте, с базовым функционалом (http, os, xml, fs, ...)

# Лабораторная работа №3

1. **Перечислите основные свойства глобальных объектов Node.js и поясните их предназначение.**

Основные глобальные объекты: global, process, buffer.

Все они доступны из любых модулей, т.е. из вообще любого места в программе без импортирования.

*Global:* Некоторые из объектов global: require, module, exports, \_\_filename, \_\_dirname.

Смысл в том, что переменные\объекты не являются полностью глобальными, а имеют видимость на уровне модуля. То есть в разных модулях можно объявить две переменные с одним именем без конфликта. Чтобы переменная\объект была видна вообще во всем приложении и всех модулях, надо объявить ее через global.varName = “value”

*Process:* Соответствует объекту процесса, который выделяется для программы. Имеет много методов для понимания того, что происходит в процессе. Есть методы для ввода\вывода в командную строку: stdin, stdout, stderr, и метод для корректного завершения приложения: exit().Также хранит инфу о среде выполнения программы (node -p "process.versions" / “process.env” / “process.release”).

Основные события process:

* beforeExit – перед тем, как в Event Loop больше не остается событий
* exit – идет после beforeExit; в общем случае процесс завершается, когда события в коде кончились; однако листенер на событие beforeExit может выполнять асинхронные вызовы. Вызовите process.exit() в любом месте программы, чтобы завершить процесс и выйти из приложения.

*Buffer:* Нужен для обработки двоичных данных. Каждый буфер соответствует некоторой памяти. Похожи на массивы целых чисел, но нельзя ресайзнуть. Можно выделить n бит под буфер и записывать туда бинарные данные.

1. **Поясните понятие «асинхронная функция».**

В асинхронной функции поток не будет блокироваться, будут отдаваться необходимые задачи, а по их завершению будет вызвана коллбэк-функция. Работа происходит в 2 фазы: отправка запроса и получение результата. То есть, процесс отдаст команду получить какие то данные и продолжит работать дальше. Асинхронность достигается процедурно за счет коллбеков, промисов (ES6) и async/await (ES8).

*For more info: https://habr.com/ru/company/ruvds/blog/424555/*

1. **Поясните понятие стандартные «системные потоки».**

Методы в глобальном объекте process: stdin(), stdout(), stderr(). Это по порядку стандартный системный поток для ввода, стандартный системный поток для вывода и стандартный системный поток для вывода ошибок.

1. **Поясните назначение функций process.nextTick,** **setImmediate, поясните в чем разница.**

Это величайшая наебка современности, потому что методы похожи, но названия должны поменяться местами.

* setImmediate() выполняется на следующем «тике» EventLoop’a, на следующей его итерации, следующей фазе (если что в ивент лупе тоже есть фазы – обычно выделяют 6)
* а nextTick() же выполняется на той же фазе EventLoop’a
* setImmediate() ставит функцию в очередь в за всеми коллбэками I/O операций в очереди событий.
* nextTick() ставит функцию в начало очереди событий, чтобы она выполнялась сразу после завершения текущей функции.
* setImmediate() выполняется на фазе Check в Ивент Лупе
* nextTick() не относится к какой то фазе, существует в отдельном микротаске. Эти микротаски имеют максимальный приоритет и будут выполняться, пока очередь микротасков не кончится. Только после этого будут выполняться макротаски (остальные фазы Ивент Лупа). То есть если запустить рекурсивно nextTick() и написать где нибудь таймер на 5сек, то таймер никогда не выполнится, ибо очередь забита микротаской nextTick()

For more info:

<https://habr.com/ru/company/lineate/blog/585050/>

https://medium.com/devschacht/event-loop-timers-and-nexttick-18579cd122e0

# Лабораторная работа №4

1. **Дайте пояснению понятию «событие программного объекта».**

Это процесс перехода объекта из одного состояния в другое, при этом, другие программные объекты могут быть извещены о том, что этот переход произошел. У каждого события есть издатель (генератор, инициатор) и подписчик (обработчик, слушатель).

1. **Объясните механизм генерации и обработки событий в C#.**

Необходим объекта типа event, а также делегат, который будет обрабатывать это событие. Для генерации – вызвать event?.Invoke();

1. **Поясните как самостоятельно реализовать механизм генерации и обработки** **событий на JS или С++.**

ну епта

1. **Какой встроенный механизм используется в Node.js для генерации и** **обработки событий. Поясните принцип его работы.**

Для этого используется стандартный модуль EventEmitter

Необходимо подключение двух модулей: events и util. Если наследоваться от EventEmitter’a через class extends или inherits. Для генерации события используется emit, а для прослушивания и обработки – on.

# Лабораторная работа №5

**Поясните принцип работы и применения функций:**

1. **setTimeout**

Является макротаской; Принимает следующие параметры:

* коллбэк, который вызовется и исполнится по истечению n миллисекунд
* число миллисекунд, через которое нужно выполнить коллбэк
* дополнительные параметры, которые могут быть переданы в коллбэк-функцию

setTimeout не гарантирует, что коллбэк будет вызван ровно через n миллисекунд. Он гарантирует только то, что функция не будет вызвана раньше, чем через n миллисекунд. Это происходит из-за того, что микротаски (например, nextTick) могут задерживать EventLoop и отодвигать таймер setTimeout дальше в очереди.

1. **setInterval**

Также является макротаской; По сути, это бесконечный цикл, который будет выполняться с определенной периодичностью. Принимает следующие параметры:

* коллбэк, который будет вызываться каждые n миллисекунд
* число миллисекунд периодичности коллбэка

setInterval также не гарантирует точного выполнения вплоть до миллисекунды по той же причине: таймер – это макротаска.

1. **unref**

Как setInterval, так и setTimeout возвращают объект типа Timeout. Мало того, что выполнение этого таймера можно отменить, передав объект Timeout в параметры функции clearTimeout или clearInterval соответственно, но можно также и изменить поведение этого таймера с помощью функций ref и unref.

Если вызвать для объекта таймера timer.unref(), то по сути это указывает, что нам похую на этот таймер и он является второстепенным. То есть этот таймер не надо проверять, чтобы понимать, завершится процесс или нет. В 5-ой лабе это помогает сделать так, чтобы даже когда у нас выполняется команды sc (выполнение коммита каждые n секунд через setInterval), то использование scTimer.unref() сделает так, чтобы я в любой момент мог прописать команду sd (дисконнект от сервера) и он отключится, несмотря на то, что у нас есть незавершенный таймер для коммитов.

1. **ref**

То же самое, что и unref, только наоборот. С помощью этой команды можно при желании отменить действие unref. На практике применяется редко.

# Лабораторная работа №6

1. **Поясните назначение npm.**

NPM – Node Package Manager – чухня, которая устанавливается вместе с нодой. Нужна для простой установки любого пакета, созданного другим разработчиком. Любой человек может опубликовать свой пакет.

1. **Перечислите команды npm, с помощью которых можно просмотреть список** **установленных пакетов.**

**npm list, npm ls, npm la, npm ll**

1. **Поясните назначение параметра –g в командах npm.**

Устанавливает пакет в глобальный репозиторий (обычно по пути установки npm Users/UserName/AppData/Roaming/npm).

Этот путь можно узнать с помощью команды **npm root -g**.

Как и с глобальными объектами, рекомендуется не загружать пакеты в глобальный репозиторий. Обычно пакеты загружаются глобально если я хочу с ними взаимодействовать через коммандную строку. Иначе, без параметра -g, пакеты устанавливаются локально, в рабочую директорию, и применяются в проектах через require или import.

1. **Перечислите команды npm,с помощью которых можно скачать пакет.**

**npm install, npm i** (allias).

После команды можно прописать необязательные параметры: путь установки, @scope, чтобы установить пакет в свою область видимости, --save чтобы также добавить запись о добавленной зависимости в package.json (после npm 5 эти записи добавляются автоматически), -g для установки в глобальный репозиторий, express@1.0.5 для установки конкретной версии, а также можно указать прямую ссылку на гитхаб пакета.

1. **Поясните назначение файла package.json.**

Основной файл кофигурации npm, нечто вроде файла манифеста. В простейшем случае, если вы просто устанавливаете пакеты, то в нем будут прописаны названия зависимостей локального репозитория npm. При создании же пакетов необходимо туда записать инфу о вашем пакете.

1. **Перечислите последовательность действий, позволяющих опубликовать** **пакет с помощью npm.**

**npm login** – сначала надо войти в свой аккаунт npm.

**npm init** – инциализируем пакет в текущей директории; создается файл package.json, который далее будет конфигурировать через командную строку.

**npm publish** – просто выгрузка пакета на ваш аккаунт, аналогия git push

# Лабораторная работа №7

1. **Поясните аббревиатуру MIME, какой организаций поддерживается допустимые MIME, в каких компонентах запросов и ответов используется MIME.**

1) MIME – Multipurpose Internet Mail Extensions — многоцелевые расширения интернет-почты – стандарт или спецификация для кодирования и форматирования информации для пересылки её по Интернету. В формат входят как текстовые данные любых кодировок, так и файлы множества форматов (картинки, видео, музыка, JSON, XML).

2) За MIME ответственна IANA (Internet Assigned Numbers Authority), все описано в RFC (конкретно в версии 6838).

3) MIME используется в заголовках, например Content-Type (со стороны сервера: то, какие данные пересылаются в ответе, чтобы клиенту было понятно, как их обрабатывать) и Accept (со стороны браузера, чтобы указать, в каком формате они предпочитают принимать данные).

Заголовок Content-Type пишется следующим образом: тип/подтип, например text/html. Тип представляет общую категорию типа данных: video, image, audio, application (неинтерпретируемый двоичный код какого-то приложения).

Подтип же определяет конкретный тип данных, определенный MIME: html, png, plain, xml, json.

И в типе, и в подтипе можно указать \* — то есть принимаются все форматы типа или подтипа. Например, \*/\* значит что браузеру ваще похуй принимаю всё что скормят.

**Основные типы и их подтипы:**

1. text – текстовые данные: html, plain, css.
2. image – графические типы: jpeg, png, gif, bmp.
3. audio – звуковые типы: mpeg (это mp3), wav, midi.
4. video – видео типы: avi, mpeg (это значит mpeg), mp4.
5. application – данные какого-то приложения: msword, pdf, zip, rtf.
6. model – данные 3D-моделей: vml, 3mf.
7. font – шрифты: ttf, otf, woff.
8. example – для написания примеров. Используется *только* в документации или примерах кода.
9. message – сообщение, включающее в себя другое сообщение.
10. multipart – данные состоят из нескольких частей разных типов: mixed, alternative (используется в e-mail).
11. **Перечислите теги HTML, интерпретация которых приводит к HTTP-запросам.**

Любые теги, которые имеют атрибут src/href, в которые можно записать адрес сервера: a, link, script, audio, video, form, img.

1. **Перечислите способы выполнения HTTP-запросов из JS-сценария.**

fetch, XmlHttpRequest.

1. **Поясните понятие «параметризованный модуль».**

Это модуль, при импорте которого можно передать в него некоторые параметры. Два варианта, как это сделать:

1) Экспортировать из модуля одну функцию, которая будет иметь параметр, и экспортировать её следующим образом:

module.exports = (parm) => new ModFunc(parm);

2) Реализовать в модуле конструктор через prototype и экспортировать не *экземпляр*, а сам *конструктор*:

Func.prototype.verifyRoles = parm = > {}

module.exports = Func;

# Лабораторная работа №8

1. **Поясните назначение заголовка Content-Type.**

Для явного определения MIME (Multipurpose Internet Mail Extensions, если кто забыл) – со стороны ответа сервера указывается заголовок Content-Type для того, чтобы клиенту было понятно, как их обрабатывать. Если этого заголовка не будет, браузер (клиент) постарается сам определить MIME тип передаваемого контента, но их реакция может быть неадекватной.

1. **Поясните назначение заголовка Accept.**

Accept также используется для явного определения MIME-типа – клиент (браузер) в заголовке Accept указывает, какие MIME-данные он предпочтёт принять и нормально обработать. Сервер выбирает одно из предложений в Accept, использует его и информирует клиента о своём выборе с помощью заголовка ответа Content-Type.

1. **Зачем нужно значение** **multipart/form-data заголовка Content-Type?**

Это – значение заголовка Content-Type (в заголовках будет указано Content-Type: multipart/form-data). Он указывается в HTML-формах при отправке через них файлов (при отправке файла через форму обязательно надо указать такое значение заголовка!). За счет значения заголовка multipart/form-data отправляемые данные будут разделены на несколько частей: одна часть – на файл, одна часть – на текстовые данные.

1. **Как с помощью тега form обеспечить значение multipart/form-data заголовка Content-Type?**

В HTML-странице это значение указывается в атрибутах тега form:

<form action="http://localhost:5000/upload" method="post" **enctype="multipart/form-data"**>. При указании этого атрибута в заголовок автоматически будет помещено Content-Type: multipart/form-data.

1. **Какое значение заголовка Content-Type отправляется тегом form в запросе по умолчанию?**

По умолчанию задается значение application/x-www-form-urlencoded. Задается кодировка как в URL: Вместо пробелов ставится +, символы вроде русских букв кодируются их шестнадцатеричными значениями (например, %D0%90%D0%BD%D1%8F вместо Аня).

1. **Где и в каком формате передаются параметры в GET-запросе?**

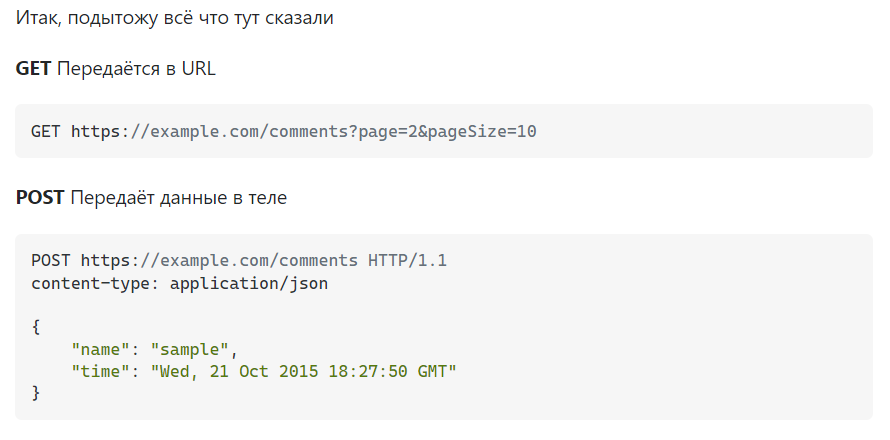
В URL – после адреса через символ вопроса указываются параметры в формате key=value, несколько параметров отделяются друг от друга через & или &&:

http://localhost:5000/parameter?x=1&y=3

1. **Где и в каком формате передаются параметры в POST-запросе?**

В теле запроса (в Postman это body – raw).

Пишется заголовок, пустая строка, и ниже тело запроса.



1. **Поясните понятие JSON?**

JSON – JavaScript Object Notation – текстовый формат обмена данными, основанный на синтаксисе объектов из JS. Произносится как джЕйсон.

1. **Поясните понятие XML?**

XML – eXtensible Markup Language – расширяемый язык разметки. Вы бля серьезно это не знаете?

# Лабораторная работа №9

**1. Чем отличается передача параметров в GET и POST запросах.**

GET – все параметры помещаются в URI

POST – параметры передаются в теле запроса

**2. Поясните структуру HTTP-ответа с вложенным файлом.**

* Используется заголовок multipart/form-data для разделения частей данных.
* Используется заголовок content-disposition, который сообщает, что контент ответа будет отображаться в браузере, как веб-страница или часть веб-страницы, или же как вложение, которое затем может быть скачано и сохранено.
* Для разделения *частей* (параметров) используется граница – boundary: можно указать любую, это необходимо для того, чтобы сервер знал, как разделить получаемые данные. После границы необходимо указать CRLF или два CRLF подряд, чтобы указать, что поля (fields) не будут передаваться.

# Лабораторная работа №10

1. **Поясните разницу между полудуплексном и дуплексным каналами данными.**

* Симплексный – передача в одну сторону: одна сторона только отправляет, а вторая только принимает, причем нет подтверждения получения передачи или возможности ответа (e.g. радио, телевидение)
* Полудуплексный – оба абонента имеют возможность принимать и передавать сообщения, НО не одновременно. У каждого из них есть приемник и ответчик (e.g. рация)
* Дуплексный – передача данных в обе стороны одновременно (e.g. мобильная связь)

1. **Какой тип канала применяется HTTP-протоколом.**

Полудуплексный.

1. **Поясните понятие WebSocket.**

Протокол передачи данных, основанный на TCP, для установки длительного соединения и обмена сообщениями между клиентом и сервером в режиме дуплексной связи. Соединение устанавливается следующим образом:

1. Клиент посылает обычный HTTP-запрос, называемый рукопожатием, с заголовками Upgrade и Connection (Upgrade: websocket)
2. Сервер решает, устанавливать WebSocket-соединение или нет, и отправит заголовок Upgrade (Upgrade: WebSocket)
3. Устанавливается соединение и HTTP заменяется на TCP/IP-соединение, далее можно отправлять данные

Данные передаются в виде фреймов с минимальным количеством служебных данных, что значительно уменьшает задержки при передаче информации.

Плюсы – высокая скорость передачи, минусы – т.к. является другим протоколом, то лишается множества функций HTTP – масштабируемости etc.

Также помимо ws: префикса схемы есть wss: префикс – аналог HTTPS

1. **Какой тип канала применяется WebSocket-протоколом.**

Дуплексный (одновременная работа; сервер может сам отправлять данные)

1. **Поясните понятия «широковещательное сообщение» (broadcast), «широковещательный сервер».**

Широковещательное сообщение – отправляемое всем узлам сети

Широковещательный сервер – принимающий запросы и отправляющий ответы всем клиентам сети

# Лабораторная работа №11

1. **Поясните понятие «TCP-порт».**

Число от 1 до 65535, указывает, какой программе предназначается пакет. Каждый открытый порт – это установленное соединение между двумя точками – соответствующей программой и узлом соединения.

1. **Поясните понятие «сетевой сокет».**

Точка для отправки и приема данных в сети; часть программного интерфейса для обеспечения передачи данных (между узлами сети или процессами). А вообще, по Шиману, сокет – это IP + port.

1. **Поясните понятие «WebSocket».**

Протокол передачи данных, основанный на TCP, для установки длительного соединения и обмена сообщениями между клиентом и сервером в режиме *дуплексной* связи.

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

Соединение устанавливается следующим образом:

1. Клиент посылает обычный HTTP-запрос, называемый рукопожатием, с заголовками Upgrade и Connection (Upgrade: websocket)
2. Сервер решает, устанавливать WebSocket-соединение или нет, и отправит заголовок Upgrade (Upgrade: WebSocket)
3. Устанавливается соединение и HTTP заменяется на TCP/IP-соединение, далее можно отправлять данные
4. **Поясните понятие «широковещательное сообщение».**

Сообщение, посылаемое всем узлам сети.

1. **Поясните принцип организации потокового ввода/вывода через WS-соединение.**
2. Создать поток WebSocketStream
3. Создать поток чтения или записи из модуля fs
4. source.pipe(destination)
5. **Поясните принцип действия и назначение механизма «ping/pong».**

Особо легкие служебные пакеты. Отправляются для того, чтобы проверить, жив вообще наш собеседник или нет. При получении ping необходимо как можно скорее отправить в ответ pong.

1. **Поясните аббревиатуру «RPC».**

Remote Procedure Call – удаленный вызов процедуры – вызов некоторой функции из другого места (другое адресное пространство, другой узел сети), причем это выглядит, как локальный вызов.

1. **Поясните принцип работы RPC-механизма, предоставляемого пакетом rpc-websockets.**
2. server.register(‘rpc-method’, <callback>);
3. если protected, то server.setAuth(login, password)
4. client.call(‘rpc-method’, params).then(res => sout(res));
5. если protected, то client.login().then(call());
6. **Поясните принцип действия и назначение механизма «subscriber/publisher».**

Publisher генерирует события – server.emit(‘event’)

Subscriber подписывается – client.subscribe(‘event’)

1. **Поясните принцип действия и назначение механизма уведомлений.**

Клиент может отправлять серверу уведомления: client.notify(‘A’)

Сервер их обрабатывает: server.register(‘A’, <callback>)

# Лабораторная работа №12

1. **Поясните понятие «файл».**

Ебать (ген. ebat’, “именновая совукопность взаимосвязанных данных, распозваваемая компьютером как единое целое”)

1. **Поясните понятие «файловая система».**

Способ хранения и упорядочивания на диске данных

1. **Перечислите типы файловых систем.**

* FAT – супер древняя
* FAT16 – очень древняя, максимальный размер файла – 2 ГБ
* FAT32 – древняя, но не настолько, как предыдущие, максимальный размер 1 файла – 4 ГБ
* NTFS – самая популярная, хотя по факту тоже древняя, максимальный размер файла – от 16 ГБ до 8 ПБ (зависит от размера кластера)

1. **Поясните понятие «поток данных».**

Поток данных (Stream) можно использовать в Node.JS в основном для работы с файлами. Если файл читается без использования потока (fs.ReadFileSync), то ожидается, пока файл не будет считан полностью. Потоки же могут считывать данные по частям и сразу начинать с ними работу, например, сразу начать отправлять порции файла клиенту.

Для отправки данных, считанных из потока, используется метод pipe().

1. **Поясните понятие «системные потоки данных».**

Наверно речь о stdin/stdout или потоках Readable/Writable, хуй знает

Если о stdin/stdout, то это стандартные унифицированные программные интерфейсы для записи/чтения данных

1. **Перечислите типы потоков данных, поддерживаемых Node.js.**

* Поток для чтения (Readable)
* Поток для записи (Writeable)
* Дуплексный поток (Duplex) – можно и читать, и записывать. Readable и Writable части работают независимо
* Трансформирующий поток (Transform) – похож на дуплексный, совмещает в себе запись и чтение; можно использовать для *трансформации* данных – шифрования, декомпресса данных, конвертирования из JSON в XML etc.

**Модули, предоставляюшие работу с потоками:**

* process.stdin, stdout, stderr – поток консоли
* fs.createReadStream() / createWriteStream() – поток для работы с файлами
* http.request() – отправляет chunk’и по потоку
* http.createServer() – данные обрабатываются порционно
* TCP-sockets – так написано в лекции
* zlib.createGzip() – сжимает данные в GZip и отправляет в поток
* net.connect() – соединение, основанное на потоке