# Лаба 9–10: Ответы на вопросы по Node.js API

## 1. Как подключить модуль (пакет) в Node.js API?

• Встроенный модуль: `const fs = require('fs');`

• Сторонний пакет (предварительно `npm install express`):

const express = require('express');

• ES‑модули (package.json `"type":"module"` или .mjs‑файл):

import express from 'express';

## 2. Какая разница между синхронными и асинхронными операциями в Node.js?

Синхронная блокирует поток, выполняется сразу и возвращает результат: `fs.readFileSync()`.

Асинхронная отправляет задачу в систему I/O, мгновенно освобождает поток, результат приходит позже через колбэк/промис: `fs.readFile()` — повышает масштабируемость, не блокируя event loop.

## 3. Что такое колбэки (callbacks) и как их использовать для асинхронного программирования в Node.js?

Колбэк — функция, передаваемая как аргумент и вызываемая после завершения асинхронной операции.

fs.readFile('file.txt', 'utf8', (err, data) => {

if (err) return console.error(err);

console.log(data);

});

Node.js следует соглашению error‑first callback: первый параметр — ошибка, второй — результат.

## 4. Как использовать промисы (promises) для обработки асинхронных операций в Node.js?

Промис — объект‑обёртка, представляющий результат, который будет доступен в будущем.

const { readFile } = require('fs').promises;

readFile('file.txt', 'utf8')

.then(data => console.log(data))

.catch(console.error);

Можно «промисифицировать» колбэк‑функции:

const util = require('util');

const read = util.promisify(fs.readFile);

## 5. Что такое асинхронные функции (async/await) и как использовать их в Node.js API?

`async` превращает функцию в промис; `await` приостанавливает её выполнение до резолва промиса.

async function run() {

try {

const data = await readFile('file.txt', 'utf8');

console.log(data);

} catch (e) { console.error(e); }

}

## 6. Как работает система обработки событий (event loop) в Node.js?

Event loop — однопоточный цикл, который вынимает задачи из очередей/фаз (timers, pending callbacks, idle, poll, check, close callbacks) и выполняет их. Асинхронные I/O операции передаются ядру, а их коллбэки ставятся в соответствующую фазу; JavaScript продолжает выполняться, пока очередь не опустеет.

## 7. Как обрабатывать параметры запроса (query parameters) в Node.js API?

В Express:

app.get('/search', (req, res) => {

const { q, page = 1 } = req.query; // ?q=js&page=2

res.json({ q, page });

});

## 8. Как парсить JSON‑данные, отправленные в теле запроса, в Node.js API?

Express‑middleware:

app.use(express.json());

app.post('/api', (req, res) => {

const data = req.body; // уже объект

res.sendStatus(200);

});

Без фреймворка — самостоятельно собирать буфер:

let body='';

req.on('data', chunk => body += chunk);

req.on('end', () => {

const obj = JSON.parse(body);

});

## 9. Как использовать базу данных в Node.js API (например, MongoDB или MySQL)?

• MongoDB (mongoose):

const mongoose = require('mongoose');

await mongoose.connect(process.env.MONGO\_URI);

const User = mongoose.model('User', new mongoose.Schema({ name:String }));

await User.create({ name:'Ivan' });

• MySQL (mysql2/promise):

const mysql = require('mysql2/promise');

const conn = await mysql.createConnection({host:'localhost', user:'root', database:'test'});

const [rows] = await conn.execute('SELECT \* FROM users');

## 10. Как обрабатывать и валидировать пользовательский ввод в Node.js API?

1. Сanitization & validation библиотеками (`express-validator`, `joi`, `zod`).

app.post('/reg',

body('email').isEmail(),

(req,res) => {

const errors = validationResult(req);

if (!errors.isEmpty()) return res.status(400).json(errors.array());

});

2. Экранирование для XSS/SQL‑инъекций; использование параметризованных запросов.

3. Лимит размеров (`express.json({limit:'10kb'})`), rate‑limiting.

## 11. Как обрабатывать заголовки запросов (request headers) в Node.js API?

В Express доступ через `req.headers` или геттеры:

app.get('/', (req,res)=>{

const ua = req.get('user-agent');

const token = req.header('Authorization');

});

## 12. Как обрабатывать файлы, отправленные вместе с запросом, в Node.js API?

Используйте middleware multer:

const multer = require('multer');

const upload = multer({ dest:'uploads/' });

app.post('/avatar', upload.single('photo'), (req,res)=>{

console.log(req.file); // info о загруженном файле

res.send('OK');

});

Для больших файлов — стриминг (например, `busboy`), S3 presigned URLs или `fs.createWriteStream` с пайпами.