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

В Node.js модули подключаются с помощью require() (CommonJS) или import (ES Modules).

## ****2.**** Какая разница между синхронными (synchronous) и асинхронными (asynchronous) операциями в Node.js?**Синхронные (Blocking):** Код выполняется последовательно, операция блокирует поток до завершения.

**Асинхронные (Non-blocking):**  
Операция выполняется в фоне, поток не блокируется. Результат обрабатывается через колбэк, промис или async/await.

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

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

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

Промис — объект, представляющий результат асинхронной операции (resolve или reject).

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

async/await —для удобной работы с асинхронным кодом.

### **Пример**

async function readData() {

try {

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

console.log(data);

} catch (err) {

console.error(err);

}

}

readData();

await можно использовать только внутри async-функций.

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

Event Loop — механизм, который обрабатывает асинхронные операции в Node.js.

### **Как работает?**

**1.Фазы Event Loop:**

**Timers** (setTimeout, setInterval).

**I/O Callbacks** (завершенные операции ввода/вывода).

**Poll** (новые события, например, HTTP-запросы).

**Check** (setImmediate).

**Close** (socket.on('close')).

**2.Блокирующие операции** (например, fs.readFileSync) могут задержать Event Loop.

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

В Express.js параметры запроса доступны через req.query

## ****8.**** Как парсить JSON-данные, отправленные в теле запроса, в Node.js API?Используйте express.json() middleware:

const express = require('express');

const app = express();

app.use(express.json());

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

const jsonData = req.body;

res.json({ received: jsonData });

});

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

### **MongoDB (с Mongoose)**

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/mydb');

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

const user = new User({ name: 'John' });

user.save().then(() => console.log('User saved'));

### **MySQL (с** mysql2**)**

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

const pool = mysql.createPool({ host: 'localhost', user: 'root', database: 'test' });

async function getUsers() {

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

return rows;

}

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

## Используйте библиотеки:

**Joi** (схемы валидации).

**Express-validator** (для Express).

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

В Express заголовки доступны через req.headers

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

multer для обработки multipart/form-data