Short Answer:

Answer the following questions with complete sentences in **your own words**. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers.

1. **What is Node.js?**

Node.js is an open-source, cross-platform JavaScript runtime environment used for executing JavaScript code outside of a web browser

2. **What are some differences between Node.js and JavaScript?**

Javascript is a programming language while Node js. is Javascript Runtime Environment.
Javscript is just for General purpose scripting while Node.js is for backend development
Javscript's JS Engine include engines such as V8, JSCore while Node JS only uses V8.
Javascript can use Web APIs while Node Js doesn't have Web API built in

3. **What is npm?**

Npm (Node package manager) is an online repository for the publishing of open-source Node.js projects. It is used for interacting with different repositories that aids in package installation, version management, and dependency management.

4. **What is CommonJS? How does it differ from ES Modules?**
    1. **How do we import files into other files in Node.js?**

CommonJS is a module formatting system that is used for structuring and organizing JavaScript code.
CommonJS is syncronous while ES modules are asyncrhonous
CommonJS uses require() to import and module.exports to export while ES modules just use import and Export.
CommonJS can import anywhere while ES modules needs to import at the beginning of the file

5. **How can you make the server automatically restart when files are modified?**

You can use Nodeman to make server automatically restart when files are modified

6. **What are some global objects in Node.js?**

Global objects in Node.js include Buffer class, console, process, global, and setImmediatte()

7. **Explain how the Node.js architecture is event-driven in terms of event emitters and other core modules.**

Node.js is event-driven because it deals with asynchronous functions. Because of this, it deals with event emitters which is a Node module that allows objects to communicate with one another and that has event-driven architecture. With event emitters, emitter objects send named events, which trigger eventlisteners that are already registered.

8. **What are streams? In Node.js, what are the different kinds of streams?**

Streams are data-handling method that are used to read or write input into output sequentially.

Writable: streams to write data.
fs.createWriteStream() lets us write data to a file.
Readable: streams to read data.
fs.createReadStream() lets us read the contents of a file.
Duplex: streams that are both Readable and Writable such as net.Socket
Transform: streams that can modify or transform the data as it is written and read.
For example, you can write compressed data and read decompressed data to and from a file with file-compression

9. What is the difference between fs module's readFile and createReadStream?

readFile	createReadStream
-reads the file into the memory before making it available to the user.	-reads the file in chunks according to a need by the user.
- slower due to read of whole file.	- faster since it brings in chunks.
-unscalable in case of too many requests as it will try to load them all at the same time.	-scalable as it pipes the content directly to the HTTP response object
-easier for nodejs to handle cleaning of memory in this case.	-not easy for memory cleaning by nodejs

10. What are the different timing or scheduling functions in Node.js?

setTimeout(): schedule code execution after a designated amount of milliseconds.
setImmediate(): execute code at the end of the current event loop cycle
setInterval(): calls a function at specified intervals (in milliseconds)

11. How does the event loop work in Node.js?

Event loop, while using callbacks and promises, waits for tasks, executes the tasks, and then waits until it receives more tasks. The event loop executes tasks from the event queue only when the call stack is empty.

12. How do you manage multiple node versions?

You use NVM(Node Version Manager) to manage multiple node version because it can add different versions of node to a specific path.

13. Explain the request & response cycle.

The request and response cycle is a cycle of request and response between clients and servers.
The Request is generated by the client and sent to the server containing the the resource the client is asking for.
The Response is what the client receives back after the Request has been fulfilled.

It is the flow of data from client to server and back. 1 request, 1 response.

14. Explain serialization vs deserialization.

Serialization: converting Object into bytes;
Deserialization: converting bytes into Objects

15. What is Express.js?

Express.js is a Node.js web application server framework, designed for building single-page, multi-page, and hybrid web applications

16. How do you build a basic server in Node.js using express?

```
// Import express with require()
const express = require('express');

// Create a new express app with express() and then save it as "app" in a variable
const app = express();

// Create a server configuration with a Port such as Port 8000
const PORT = 8080;

// Create a Route for the app with the get() method that has response, and respond in the
parameter
app.get('/', (req, res) => {
  res.send('Hello World');
});

// Make the serve listen to request with the listen () method to run the Port
app.listen(PORT, () => {
  console.log(`Server running at: http://localhost:${PORT}/`);
});
```

17. What is middleware?

Middle ware are functions that are executed in between the server receiving the request (req) and responding with a response (res). Middleware functions are often responsible for parsing requests, logging, and handling errors.

Ex. Function(req,res,next){}

18. What are ports? Which one is reserved for HTTP and HTTPS?

Ports are communication endpoint for computers connected to a network with an IP Adress..
Port 80 is reserved for HTTP. HTTPS port is for port 443

19. What is the difference between response.send() and response.write() in express?

A Response.send() can be called only once while response.write() can be called multiple times followed by a response.

20. Explain query strings vs route parameters.

Query strings/parameters allow you to pass optional parameters to a route such as pagination information

Route parameters are variables derived from named sections of the URL and are essential to determining route, whereas query parameters are optional.

21. Explain MVC and the file structure in express.

MVC(Model View Controller) is an Architectual pattern with three main logical component: Model, View, and Controller

Model: Logic relationg to database;
View: Logic relating to Frontend UI
Controller: Logic dealing with HTTP request and response.

Coding Questions:

Use HTML/CSS/JS to solve the following problems. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it would be easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. **To-Do App (Express.js)**
   Implement an express server that handles requests for a basic to-do app and interacts with the file system. Be sure to test your routes with Postman and submit screenshots of the server responses.
   - Feel free to implement the front-end (HTML/CSS/JS) if you have time, but it is **not required**.
   - The to-do app should include the following route paths & feature requirements:

   1. `POST: /todo/{filename}`
      The incoming POST request url contains the name of a local .json file that contains all the to-do items. (**hint: route parameters**)
      - The request body should contain JSON data that corresponds to a to-do item: **title**, **description**, **status** (In Progress, Completed), and **priority** (Low, Medium, High). The

server should generate an additional property, **“timestamp”**, using the current timestamp.
- If no file exists, create a new .json file with this name.
- Add the to-do item to the specified json file. Then respond with the new json file contents and appropriate status code.
- If there is an error, respond with the appropriate error, message, and status code.

2. `GET: /todo/{filename}`
   The incoming GET request url contains the name of a local .json file that contains all the to-do items. It is up to you if you want to use query strings or route parameters.
   - If the specified file exists, respond with the file contents.
   - If no file exists or there is an error, respond with the appropriate error, message, and status code.

3. `PUT: /todo/{filename}`
   The incoming PUT request url contains the name of a local .json file that contains all the to-do items. (**hint: route parameters**)
   - The request body should contain JSON data with the to-do item’s title and new title, description, status or priority.
   - If the specified file exists, modify the corresponding to-do item based on the request body. Respond with the new json file contents and appropriate status code.
   - If no file exists or there is an error, respond with the appropriate error, message, and status code.

4. `DELETE: /todo/{filename}`
   The incoming DELETE request url contains the name of a local .json file that contains all the to-do items. (**hint: route parameters**)
   - The request body should contain JSON data with the to-do item’s title.
   - If the specified file exists, remove that to-do item. Respond with the new json file contents and appropriate status code.
   - If no file exists or there is an error, respond with the appropriate error, message, and status code.

Paired Programming:

Use JS to solve the following problems with your partner. Please remember to label who was acting as the driver and navigator for each problem and write down your feedback on their performance. **Feedback will be confidential**.

Leetcode #205: Isomorphic Strings
Leetcode #206: Reverse Linked List