Experiment No 10

Aim :- Write a program "Hello world" using http module of Node.js Demonstrate REPL in Node.js

Theory :- The HTTP module in Node.js allow us to create a basic web server that can listen for HTTP requests and respond to them.

How the program works :-

- http.createServer() : This method creates a new web server. it accepts a callback function that will be executed every time a request is received by the server.

- res.writeHead() :- This method sets the status code and Headers for the response. In this case it sends a 200 ok status with a content type of text/plain

- res.write() :- This method is used to send the body of the response, which is "Hello, world!".

- res.end() : This method signals the end of the response.

- server.listen(3000) : This tells the server to listen on port 3000. When the server is ready, it logs a message to the console indicating that it is running.

REPL in Node.js :- REPL stands for Read-Eval-Print Loop. It is an Interactive shell provided by Node.js where you can enter commands, have them evaluate and see the output immediately. This makes it useful for testing and debugging.

The Repl performs the following tasks:

I) Read: Reads user input and parses it into Javascript data structure.

II) Eval: Evaluates the parsed data structure.

III) print: prints the result of the evaluation.

IV) Loop: loops back to read more input.

REPL Commands :→

I) .exist :- Exit the REPL
II) .help :- Show REPL Commands
III) .save filename :- Save the current REPL session to a file→

IV) .load filename : Load a file into REPL for execution.

Conclusion:→ Hence we have successfully implemented a program "HelloWorld" using http module of Node.js.

Experiment No : 11

Aim:- Write a program which will perform file operation in Node.js using file system module.

Theory:- The fs (file system) module in Node.js provides an API to interact with the file system. it allows you to perform file operation like reading writing, updating, deleting and renaming files.

Explanation of file operations.

1) fs.writeFile(): creates a new file and writes data to it.

→ If the file already exists, it overwrites the content.

→ Asynchronous and non-blocking.

→ Arguments :- The path of the file, content to write, callback function that handles any errors.

2) fs.appendFile :- Adds data to an existing file without overwriting the content.

→ If the file does not exist, it creates a new one.

→ Asynchronous and non-blocking.

→ Arguments :- The file path, content to append, callbacks function that handles any errors.

3) fs.readFile():- Read the content of a file.
→ Asynchronous and non-blocking.
→ Arguments :- The file path.
→ The encoding ('utf8' to read the file as a string).
Callback function to handle the data or error.

4) fs.rename():- You can change the name of a file using fs.rename(). This is particularly useful for renaming uploaded files or organizing files in different categories.

5) fs.unlink:→ The fs.unlink() method delete a file from the file system. It's important for maintaining disk space and removing unnecessary files.

Conclusion:→ Hence, we have sucessfully implemented a program which will perform file operation in Node.js using file system module.

Experiment No 12

Aim :→ Write a program which demonstrates session handling in Express JS.

Theory :→ Express Js is a minimal and flexible web application framework for Node.Js that provide a robust set of features for building web and mobile applications. it simplifies the process of creating server side application, by offering built-in middleware routing, and other utilities for handling HTTP request and response.

Session Handling in Express.js :→ session Handling is a mechanism used to maintain state across multiple requests in a web application. since HTTP is a stateless protocol it does not keep track of previous interactions between the client and server. sessions allow the server to store user data for the duration of a user's interaction with a website or until the session is terminated.

In Express.Js, session handling is commonly done using the express-session middleware. This middleware allows you to store data on the server and gives each client a unique session ID stored in cookies.

Key concept of session Handling:→

I) Session ID:→ A unique identifier stored in a cookie on the client side and sent with each request to the server.

II) Session store:- Stores session data on the server-side By default, express, session store data in memory

III) Session Expiry:→ Sessions can be set to expire after a specified period, allowing for session management & improved security.

Conclusion:→ Hence, we have successfully implemented a program which demonstrates session handling in Express.Js.