Q1:

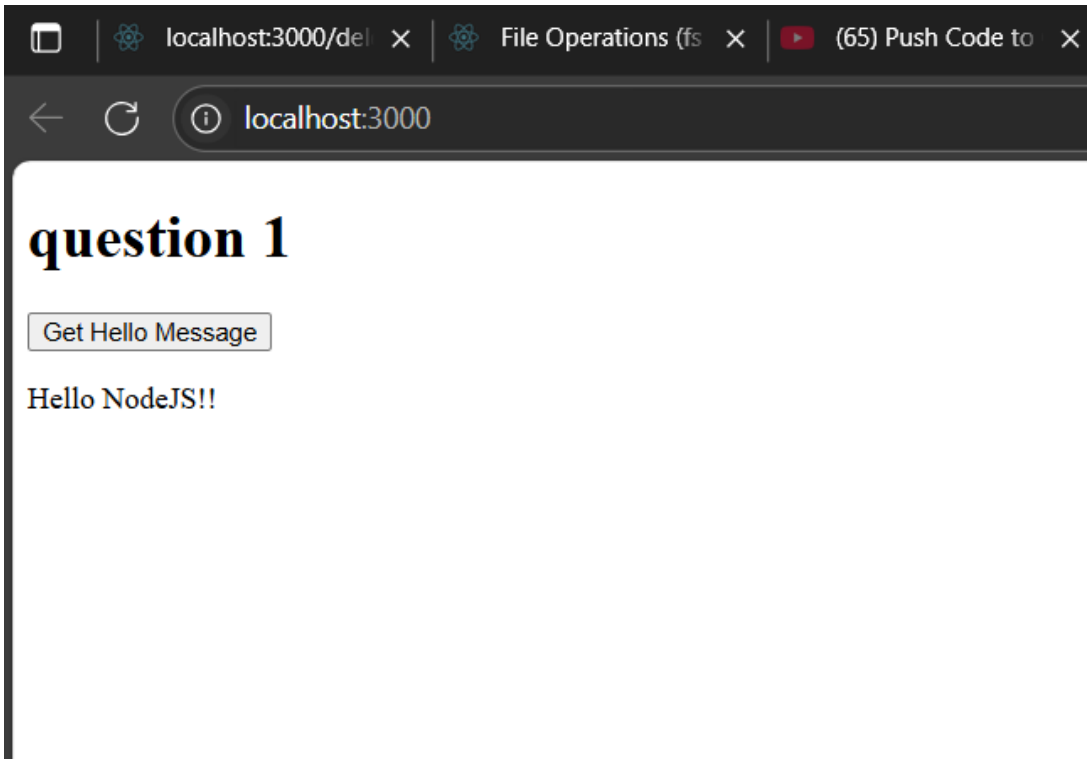
- index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>

    function getHello() {
      $.ajax({
        url: '/gethello',
        type: 'GET',
        success: function(response) {
          document.getElementById("helloMessage").innerText = response;
          //  alert("hello node");
        },
        error: function(error) {
          console.log('Error:', error);
        }
      });
    }
  </script>
</head>
<body>
  <h1>question 1</h1>
  <button onclick="getHello()">Get Hello Message</button>
  <p id="helloMessage"></p>
```

```
</body>
</html>
```

- Server.js:

```
const express = require('express');
const app = express();
const port = 3000;

app.use(express.static('static'));

app.get('/gethello', (req, res) => {
  res.send('Hello NodeJS!!');
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```



Q2:

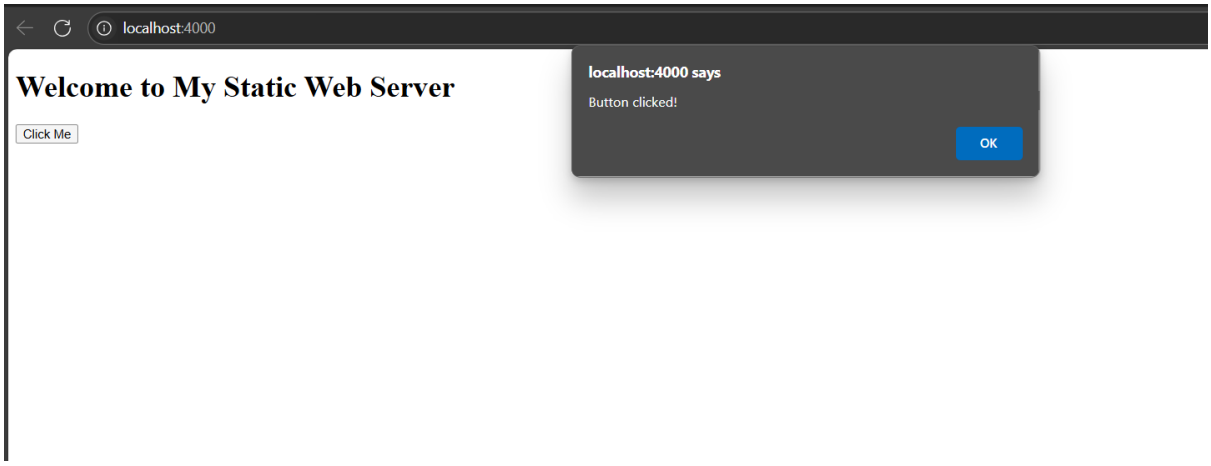
- static/index.js:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Static Web Server</title>

</head>
<body>
  <h1>Welcome to My Static Web Server</h1>
  <button id="clickButton">Click Me</button>

  <script>
    document.getElementById("clickButton").addEventListener("click", function () {
      alert("Button clicked!");
    });
  </script>
</body>
</html>
```

- Server.js:

```
const express = require('express');
const path = require('path');
const app = express();
const port = 4000;
```

```
app.use(express.static('static'));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'static', 'index.html'));
});

// Start the server
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```



Q3:

Server.js:

```
// Import the readline module
const readline = require('readline');

const today = new Date();


const y = today.getFullYear();
```

```
const m = today.getMonth() + 1;
const d = today.getDate();
const dmy = `${d.toString().padStart(2, '0')}-${m.toString().padStart(2, '0')}-${y}`;

const chatbotResponses = {
  "hello ai": "hey what's your day going on today,if you need any help, please tell me...",
  "gm": "good morning, krunal:)",
  "date?": `today date is ${dmy}`,
  "wather?": `curently i dont able to show you tempreature`,
  "default": "Sorry, I didn't quite understand that. Can you please rephrase your question? Try asking about workouts, diet, or general fitness tips."
};

function getResponse(query) {
  const normalizedQuery = query.toLowerCase();
  for (let key in chatbotResponses) {
    if (normalizedQuery.includes(key)) {
      return chatbotResponses[key];
    }
  }
  return chatbotResponses["default"];
}

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

function startChatbotCLI() {
  console.log("my ai:");
  console.log("Ask me anything. Type 'exit' to quit.\n");
```

```
  rl.question('You: ', (query) => {
    if (query.toLowerCase() === 'exit') {
      console.log("Goodbye!");
      rl.close();
      return;
    }

    const response = getResponse(query);
    console.log(`Bot: ${response}\n`);
    startChatbotCLI();
  });
}

startChatbotCLI();
```

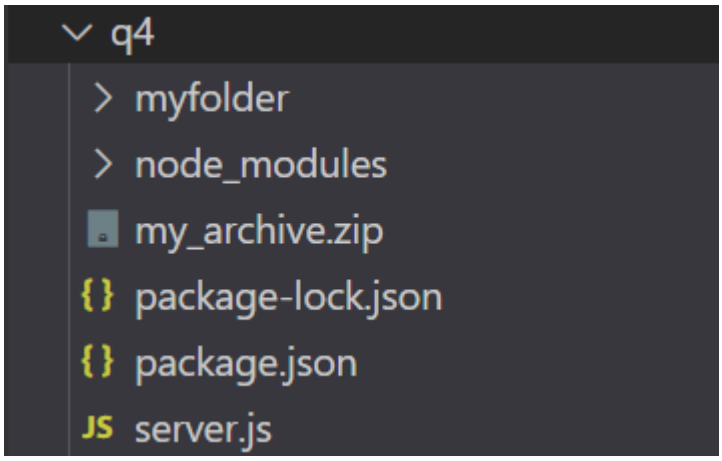
```
my ai:
Ask me anything. Type 'exit' to quit.

You: hello ai
Bot: hey what's your day going on today,if you need a

my ai:
Ask me anything. Type 'exit' to quit.

You: gm
Bot: good morning, krunal:)

my ai:
Ask me anything. Type 'exit' to quit.

You: date
Bot: Sorry, I didn't quite understand that. Can you p
king about workouts, diet, or general fitness tips.

my ai:
Ask me anything. Type 'exit' to quit.

You: date?
Bot: today date is 21-07-2025

my ai:
Ask me anything. Type 'exit' to quit.

You:
```

Q4:

Server.js:

```
const fs = require('fs');
const path = require('path');
const archiver = require('archiver');
const yauzl = require('yauzl'); // For unzipping
const mkdirp = require('mkdirp');

const mypath = './myfolder';
const zipname = 'my_archive.zip';
const outputZipPath = path.join(__dirname, zipname);

async function createZipArchive(source, output) {
  return new Promise((resolve, reject) => {
    if (!fs.existsSync(source)) {
      return reject(new Error(`Source folder does not exist: ${source}`));
    }

    // const archive = archiver('zip', {
    //     zlib: { level: 1 }
    // });

    const outputStream = fs.createWriteStream(output);

    outputStream.on('close', () => {
      console.log(`\nZip archive created successfully!`);
      console.log(`Output file: ${output}`);
      resolve();
    });
  });
```

```
}
async function main() {
  console.log(`Attempting to zip folder: ${mypath}`);
  try {
    await createZipArchive(mypath, outputZipPath);
    console.log('Zip creation process finished.');
  } catch (error) {
    console.error('Error creating zip archive:', error.message);
  }
}

main();
```



Q5:

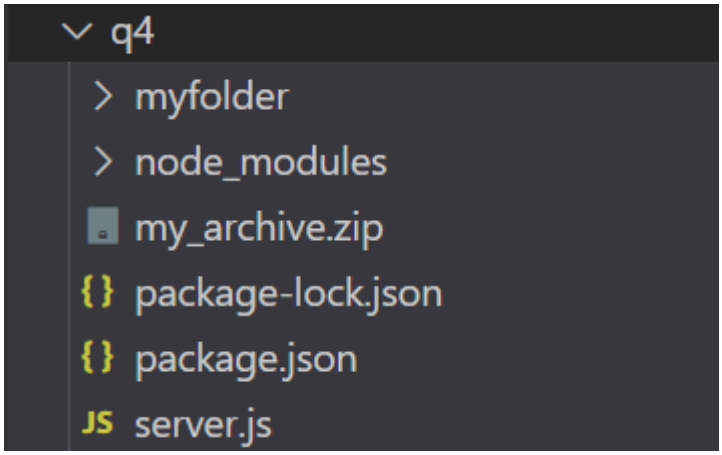
Server.js:

```
const fs = require('fs');
const path = require('path');
const yauzl = require('yauzl'); // For unzipping
const mkdirp = require('mkdirp');
// --- Function to unzip an archive ---
async function unzipArchive(zipFilePath, destinationPath) {
```

```
return new Promise((resolve, reject) => {
  console.log(`\nAttempting to unzip ${zipFilePath} to ${destinationPath}`);

  // Check if the zip file exists
  if (!fs.existsSync(zipFilePath)) {
    return reject(new Error(`Zip file does not exist: ${zipFilePath}`));
  }

  yauzl.open(zipFilePath, { lazyEntries: true }, (err, zipfile) => {
    if (err) {
      return reject(err);
    }

    zipfile.on('entry', (entry) => {
      const entryPath = path.join(destinationPath, entry.fileName);

      // If it's a directory (ends with '/'), create it
      if (/\/$/.test(entry.fileName)) {
        mkdirp(entryPath)
          .then(() => zipfile.readEntry())
          .catch(reject);
      } else {
        // If it's a file, ensure its parent directory exists and then extract
        mkdirp(path.dirname(entryPath))
          .then(() => {
            zipfile.openReadStream(entry, (err, readStream) => {
              if (err) {
                return reject(err);
              }
              const writeStream = fs.createWriteStream(entryPath);
              readStream.pipe(writeStream);
```

```
                    writeStream.on('finish', () => zipfile.readEntry());
                    writeStream.on('error', reject);
                  });
                })
                .catch(reject);
            }
        });

        zipfile.on('close', () => {
            console.log('Unzipping completed successfully!');
            resolve();
        });

        zipfile.on('error', reject);

        zipfile.readEntry(); // Start reading entries
    });
  });
}

async function main() {
  if (fs.existsSync(unzipDestinationPath)) {
    console.log(`Removing existing unzipped folder: ${unzipDestinationPath}`);
    fs.rmSync(unzipDestinationPath, { recursive: true, force: true });
  }

  try {
    await unzipArchive(outputZipPath, unzipDestinationPath);
    console.log('Unzip process finished.');
  } catch (error) {
    console.error('Error unzipping archive:', error.message);
```

```
    }
}

main();
```



Q6:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Delete File using Button</h2>
  <button onclick="deleteFile()">Delete newfile.txt</button>
    <script>
        // Function to delete the file
        function deleteFile() {
            fetch('/delete')
```

```
        .then(res => res.text())
        .then(data => {
          document.getElementById('result').innerText = data;
          alert(data);
        })
        .catch(err => {
          document.getElementById('result').innerText = 'Error deleting file';
          console.error(err);
        });
        }

        // Call the deleteFile function when the script loads
        deleteFile();
    </script>

</body>
</html>
```

Server.js:

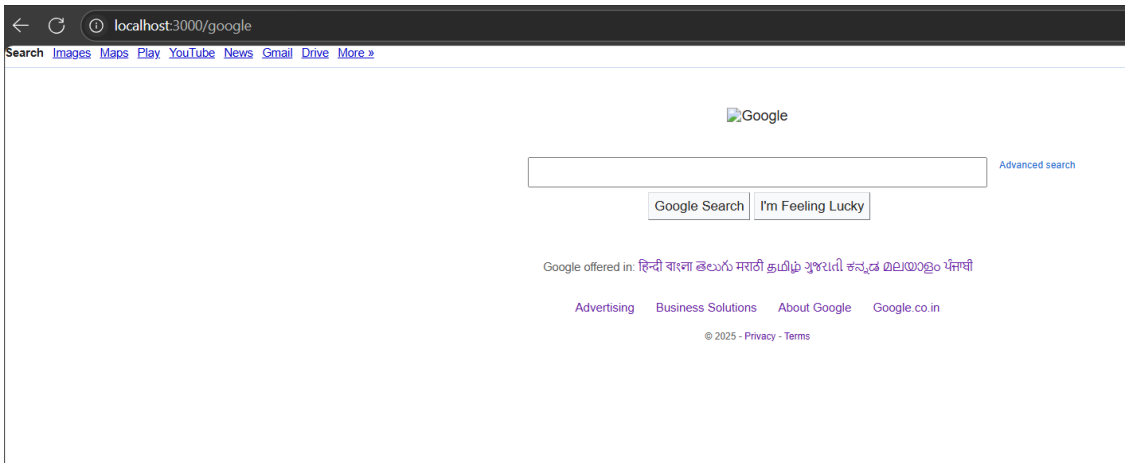
```
const express = require('express');
const fs = require('fs');
const path = require('path');

const app = express();
const PORT = 3000;

// Route to serve index.html
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'index.html'));
```

```
});

app.get('/delete', (req, res) => {
  fs.unlink('newfile.txt', (err) => {
    if (err) {
      console.error(err);
      res.status(500).send('Internal Server Error');
      return;
    }
    console.log('File deleted');
    res.send('File newfile.txt deleted');
  });
});

app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

Q7:

Server.js:

```
const express = require('express');
const fetch = require('node-fetch'); // works with v2
const app = express();
const PORT = 3000;

app.get('/google', async (req, res) => {
  try {
    const response = await fetch('https://www.google.com');
    const html = await response.text();
    res.send(html);
  } catch (error) {
    console.error('Error fetching Google:', error.message);
    res.status(500).send('Failed to fetch Google homepage');
  }
});
app.listen(PORT, () => {
  console.log(`Server is running at http://localhost:${PORT}`);
});
```



Q8:

Package.json:

```
{
  "name": "q8",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "server": "node server.js",
    "test": "echo \"No tests specified\" && exit 0",
    "greet": "echo Hello, world!",
    "lint": "eslint .",
    "start:dev": "nodemon server.js"

  }
}
```

```
PS E:\node project\SEM7-105\SEM7-105\105_krunal_701_A1\Q8> npm run lint

  > q8@1.0.0 lint
  > eslint .

  'eslint' is not recognized as an internal or external command,
  operable program or batch file.
PS E:\node project\SEM7-105\SEM7-105\105_krunal_701_A1\Q8>
```

Q9:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <title>File Operations (fs + Express)</title>

</head>
<body>
  <h2>File Operations with Express and fs</h2>

  <button onclick="callRoute('write')">Write File</button>
  <button onclick="callRoute('append')">Append to File</button>
  <button onclick="callRoute('read')">Read File</button>
  <button onclick="callRoute('rename')">Rename File</button>
  <button onclick="callRoute('exists')">Check if File Exists</button>
  <button onclick="callRoute('delete')">Delete File</button>

  <div id="output"></div>

  <script>
    function callRoute(route) {
      fetch(`/${route}`)
        .then(res => res.text())
        .then(data => {
          document.getElementById('output').innerText = data;
        })
        .catch(err => {
          document.getElementById('output').innerText = ' Error: ' + err;
        });
    }
  </script>
</body>
</html>
```

Server.js:

```
const express = require('express');
const fs = require('fs');
const path = require('path');

const app = express();
const PORT = 3000;

const filePath = path.join(__dirname, 'demo.txt');
const renamedPath = path.join(__dirname, 'renamed_demo.txt');

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});
// Write a file
app.get('/write', (req, res) => {
  fs.writeFile(filePath, 'This is a demo file.\n', (err) => {
    if (err) return res.status(500).send('Error writing file');
    res.send(' File written successfully');
  });
});

// Append to file
app.get('/append', (req, res) => {
  fs.appendFile(filePath, 'Appending new line...\n', (err) => {
    if (err) return res.status(500).send('Error appending file');
    res.send(' Text appended');
  });
});
```

```
// Read file
app.get('/read', (req, res) => {
  fs.readFile(filePath, 'utf8', (err, data) => {
    if (err) return res.status(500).send('Error reading file');
    res.send(`<pre>${data}</pre>`);
  });
});

// Rename file
app.get('/rename', (req, res) => {
  fs.rename(filePath, renamedPath, (err) => {
    if (err) return res.status(500).send('Error renaming file');
    res.send(' File renamed to renamed_demo.txt');
  });
});

// Check if file exists
app.get('/exists', (req, res) => {
  const exists = fs.existsSync(filePath) || fs.existsSync(renamedPath);
  res.send(` File exists: ${exists}`);
});

// Delete file
app.get('/delete', (req, res) => {
  const target = fs.existsSync(filePath) ? filePath : renamedPath;
  fs.unlink(target, (err) => {
    if (err) return res.status(500).send('Error deleting file');
    res.send(' File deleted');
  });
});
```

```
app.listen(PORT, () => {
   console.log(` Server running at http://localhost:${PORT}`);
});
```

Q10:

Index.js:

```
global.myGlobalValue = 'Accessible everywhere!';

console.log('--- Node.js Global Objects Demo ---');
console.log('My global value:', global.myGlobalValue);

console.log('Node version:', process.version);
console.log('Process ID:', process.pid);

console.log('Current directory (__dirname):', __dirname);
console.log('Current file (__filename):', __filename);

setTimeout(() => {
  console.log('This message is displayed after 2 seconds');
}, 2000);
```

```
console.log('Command-line arguments:', process.argv);
```