Assignment: Node.js Modules and File System

Question 1: Output in ES Module and CommonJS

Code:

```
import fs from "fs"; // or const fs = require('fs');

process.nextTick(() => console.log('nextTick 1'));
Promise.resolve().then(() => console.log('promise 1'));
setImmediate(() => { console.log('setImmediate 1') });
setTimeout(() => console.log('setTimeout 1'), 0);
fs.readFile('./files/input.txt', "utf-8", (err, data) => {
    if (err) console.log('there is an error. can not read from file');
    else console.log(data);
});
```

Expected Output Order:

1. nextTick 1
2. promise 1
3. (File content OR error)
4. setTimeout 1
5. setImmediate 1

Note: I/O operations may complete before or after timers depending on system performance.

Question 2: HTTP Server Implementation

Below is a complete Node.js HTTP server implementation using the `http` and `fs` modules:

```
import http from 'http';
import fs from 'fs';
import path from 'path';

const server = http.createServer((req, res) => {
  const { url, method } = req;

  if (method === 'GET') {
    if (url === '/' || url === '/home') {
      res.writeHead(200, { 'Content-Type': 'text/plain' });
      res.end('Welcome to my website');
    } else if (url === '/image') {
      const imgPath = path.join('./files/image.jpg'); // path to image
      fs.readFile(imgPath, (err, data) => {
        if (err) {
          res.writeHead(500);
          res.end('Image not found');
        } else {
          res.writeHead(200, { 'Content-Type': 'image/jpeg' });
          res.end(data);
        }
      });
```

```
  } else if (url === '/pdf') {
    const pdfPath = path.join('./files/sample.pdf');
    fs.readFile(pdfPath, (err, data) => {
      if (err) {
        res.writeHead(500);
        res.end('PDF not found');
      } else {
        res.writeHead(200, { 'Content-Type': 'application/pdf' });
        res.end(data);
      }
    });
  } else if (url === '/about') {
    const txtPath = path.join('./files/about.txt');
    fs.readFile(txtPath, 'utf8', (err, data) => {
      if (err) {
        res.writeHead(500);
        res.end('Text not found');
      } else {
        res.writeHead(200, { 'Content-Type': 'text/plain' });
        res.end(data);
      }
    });
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('404 Not Found');
  }
} else {
```

```
    res.writeHead(405);
    res.end('Method Not Allowed');
  }
});

server.listen(3000, () => {
  console.log('Server running at http://localhost:3000');
});
```

Question 3: Reading Files in Node.js

1. fs.readFileSync(path, encoding)
   - Synchronous and blocking
   - Ideal for small files or scripts that run once

2. fs.readFile(path, encoding, callback)
   - Asynchronous with callback
   - Non-blocking, general purpose

3. fs.promises.readFile(path, encoding)
   - Asynchronous with Promise
   - Used with async/await

4. fs.createReadStream(path)
   - Streams data in chunks
   - Best for large files or continuous reading

Example:

```
const stream = fs.createReadStream('./file.txt', 'utf8');

stream.on('data', chunk => console.log(chunk));

stream.on('end', () => console.log('DONE'));
```

This completes my assignment.