Lab13

1. **Question1:**

commonJS module

nextTick 1
promise 1
setTimeout 1
setImmediate 1
data from input.txt file

ES module

promise 1
nextTick 1
setImmediate 1
setTimeout 1
data from input.txt file

2. **Question3:**

fs.readFileSync():

- Blocks the entire thread until the file is read.
- Should be avoided in production web servers.

Example:

```
import fs from 'fs';
const data = fs.readFileSync('./files/about.txt', 'utf-8');
console.log(data);
```

fs.readFile()

- Non-blocking
- Old-style Node callback, still widely used

Example:

```
import fs from 'fs';
fs.readFile('./files/about.txt', 'utf-8', (err, data) => {
  if (err) return console.error('Error:', err);
  console.log(data);
});
```

fs.promises.readFile()

- Promise-based, perfect for async/await
- Preferred in modern Node projects

Example:

```
import { readFile } from 'fs/promises';
```

```
try {
  const data = await readFile('./files/about.txt', 'utf-8');
  console.log(data);
} catch (err) {
  console.error('Error:', err);
}
```

fs.createReadStream()

- Best for large files
- Uses less memory
- You can pipe it to HTTP responses: stream.pipe(res);

```
Example:
import fs from 'fs';
const stream = fs.createReadStream('./files/about.txt', 'utf-8');
stream.on('data', chunk => {
  console.log('CHUNK:', chunk);
});
stream.on('end', () => {
  console.log('File reading done.');
});
```