Question3: practice the fs.readFileSync(), fs.readFile(), fspromises.readFile(), and fs.createReadStream() methods. What are the differences?

fs.readFileSync() — Synchronous

- Blocking: execution waits until the entire file is read.
- Loads the full file into memory.

Example usage:

```
import fs from "fs";
const data = fs.readFileSync("input.txt", "utf-8");
console.log("data", data);
console.log("end");

// Output:
data [input file content]
end
```

fs.readFile() — Asynchronous with Callback

- Non-blocking: uses a callback function.
- Loads the full file into memory.

Example usage:

```
import fs from "fs";
fs.readFile("input.txt", "utf-8", (err, data) => {
  console.log("data", data);
});
console.log("end");

// Output:
end
data [input file content]
```

fs.promises.readFile() — Asynchronous with Promise

- Non-blocking: returns a Promise.
- Loads the full file into memory.

Example usage:

```
import fs from "fs";
async function getData() {
  const data = await fs.promises.readFile("input.txt", "utf-8");
  console.log("data", data);
}
```

```
getData();
console.log("end");

// Output:
end
data [input file content]
```

fs.createReadStream() — Streaming

- Non-blocking and memory-efficient.
- Reads file in chunks, emits 'data' events.
- Ideal for large files.

Example usage:

```
import fs from "fs";
const stream = fs.createReadStream("input.txt");
stream.on("data", (chunk) => {
  console.log(chunk.toString());
});
stream.on("end", () => {
  console.log("done reading");
});
console.log("end");

// Output:
end
[chunk]
done reading
```