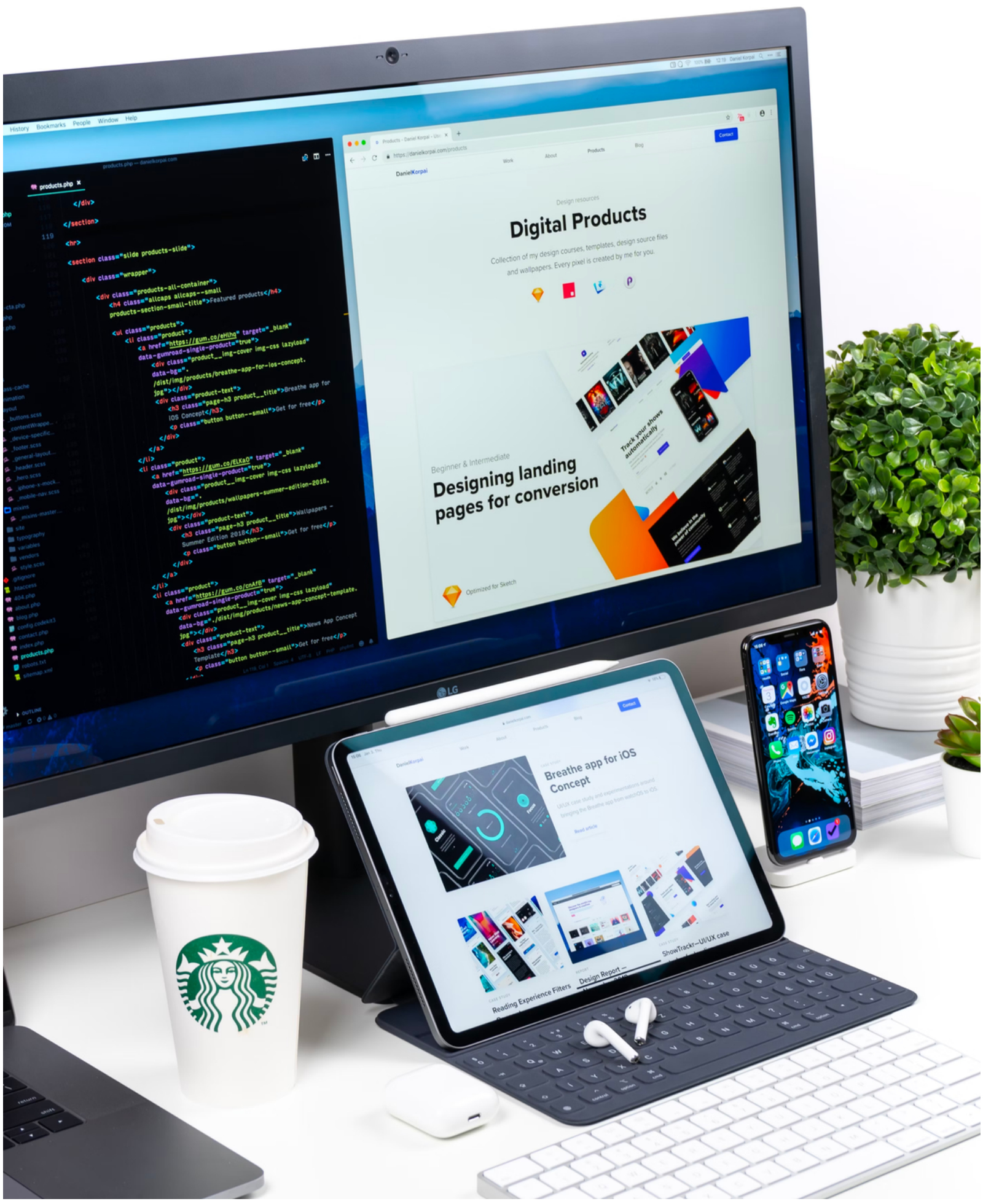
Lección 2: Escritura de archivos, logs



Tiempo de ejecución: 4 horas

Planteamiento de la sesión

Materiales:

- https://github.com/daduque/innovador_project.git
- https://nodejs.org/docs/v20.11.1/api/path.html#path
- File system | Node.js v21.7.1 Documentation
- https://nodejs.org/api/fs.html#file-system-flags
- Date - JavaScript | MDN
- Node.js - Request Object

En la lección anterior se presentaron los módulos path y file system, los cuales nos permiten interactuar con rutas y archivos en nuestro proyecto, ahora que tenemos una ruta, un controlador y una vista podemos comenzar a hacer el registro de logs, por ejemplo que se ingresó a la ruta raíz en una fecha determinada.

Así como ahora no es mucho lo que podemos guardar en los logs por las pocas funcionalidades desarrolladas hasta el momento, el funcionamiento es el mismo a la hora de querer guardar otro tipo de información .

Para la solución a este reto vamos a trabajar en la rama step3, que tendrá ya la solución a la hora de revisarla, para cambiar a la rama el comando es “git checkout step3”

Creando el archivo

Vamos entonces a crear un bloque de código dentro del controlador que permita escribir en un archivo de logs previamente creado, esto lo podemos hacer con el método del FS llamado appendFileSync(), sin embargo, puede ser interesante revisar el método llamado createWriteStream(), que permite crear el archivo si no existe e incluso el método mkdir(), que permite crear el directorio.

Como lo indica la documentación del método Node.js fs.createWriteStream() Method - GeeksforGeeks

fs.createWriteStream(path, options)

Recibe un path o ruta para la creación del archivo, finalizando con el nombre del archivo, por ejemplo, como en nuestro caso fs.createWriteStream(path.join('logs', 'general_logs.txt'), …)

y como segundo parámetro, un objeto con las opciones, o flags File system | Node.js v21.7.1 Documentation que indica de qué forma procede con el archivo, por ejemplo 'a' para crear si no existe o 'w', que es su valor por defecto, para crear o sobreescribir. Para nuestro caso queremos que intente crear el archivo solo si no existe, pero que no sobreescriba nada.

```
fs.createWriteStream(path.join('logs', 'general_logs.txt'), { flags: 'a' });
```

Por ahora entonces el método index del controlador se vería así:

```
const BooksController = {
    index: (req, res) => {
        //write the log message to the general_logs.txt file
            const log = fs.createWriteStream(path.join('logs', 'general_logs.txt'), {
flags: 'a' });
        //read the file books.json from data directory and get the data
        let data = fs.readFileSync(path.join('data', 'books.json'), 'utf8');
        //parse the data to convert it into an array of objects
        let books = JSON.parse(data);
        //render the index view and pass the books data to the view
        res.render('index', { books, title: 'Bootcamp Book Store' });
    }
}
```

Para que al ejecutar el aplicativo funcione correctamente, es necesario que se haya creado previamente el directorio llamado logs, ya que no estamos haciendo uso del mkdir para crear si no existe.

Al ejecutar el programa, debe aparecer un nuevo archivo llamado general_logs.txt dentro del directorio logs.

Pero hasta ahora no hemos escrito nada dentro del fichero, no hemos creado el mensaje. Definimos el mensaje como una fecha con hora, un texto y la info de la ruta, entonces vamos a dividir ese ejercicio en 3 partes.

Lo primero es obtener la fecha y hora del sistema, para ello JS cuenta con una serie de métodos relacionado con la clase Date() que podemos revisar en la documentación Date - JavaScript | MDN, sin embargo nos interesa solo uno llamado toLocaleString(), que nos devuelve la fecha y la hora, algo como '14/2/2024, 12:21:59 p. m.', el código entonces se vería como:

```
const dateTime = new Date().toLocaleString();
```

No es necesario generar una variable para obtener la información pero para el caso práctico comenzaremos por allí.

Ya estaría resuelta la primera parte, ahora es necesario generar el mensaje, es solo un texto que indica lo que queremos decir

```
const message = 'ingreso en la ruta '
```

Y así solo faltaría la tercera parte que es obtener la ruta, esta info la encontramos en el req o request, en un atributo llamado url Node.js - Request Object que podemos concatenar con el mensaje para obtener algo como:

```
const message = 'ingreso en la ruta ' + req.url;
```

Verificar el mensaje

Ahora ya tenemos las 3 partes completas, solo restaría escribir el mensaje completo en el archivo y verificar que cada que ingresemos a la ruta, el registro sea guardado. Podemos hacer un paso previo y es imprimir el mensaje completo para verificar que haya quedado bien, para ello podemos hacer uso del console.log:

```
console.log(dateTime + ' - ' + message);
```

el resultado debe verse por la consola de VS Code como

Ahora si que verificamos que todo esté correcto, pasamos a la escritura en el archivo

Escribiendo un registro

ya sabemos que es con el método appendFileSync(), sabemos el path al archivo y el mensaje a guardar, al ponerlo todo junto quedaría algo como:

fs.appendFileSync(path.join('logs', 'general_logs.txt'), dateTime + ' - ' + message + '\n');

Agregando un salto de línea al final del archivo.

Y al revisar el archivo de logs se debería ver algo como:

15/2/2024, 12:21:59 p. m. - ingreso en la ruta /
15/2/2024, 12:22:58 p. m. - ingreso en la ruta /
15/2/2024, 12:15:20 p. m. - ingreso en la ruta /
15/2/2024, 12:37:14 p. m. - ingreso en la ruta /

y con cada recarga de la página, debe aparecer un nuevo registro.

Al final el controlador en su método index se vería así:

```
    index: (req, res) => {
        const message = 'ingreso en la ruta ' + req.url;
        const dateTime = new Date().toLocaleString();
        // console.log(dateTime + ' - ' + message);
        //write the log message to the general_logs.txt file
        fs.createWriteStream(path.join('logs', 'general_logs.txt'), { flags: 'a' });
            fs.appendFileSync(path.join('logs', 'general_logs.txt'), dateTime + ' - ' +
message + '\n');
        //read the file books.json from data directory and get the data
        let data = fs.readFileSync(path.join('data', 'books.json'), 'utf8');
        //parse the data to convert it into an array of objects
        let books = JSON.parse(data);
        //render the index view and pass the books data to the view
        res.render('index', { books, title: 'Bootcamp Book Store' });
    }
```