Práctica 3. Uso del sistema de archivos y argumentos con NodeJS

Es hora de comenzar a crear tu primera aplicación Node.js, utilizando el sistema de archivos y los argumentos de la línea de comandos para crear una aplicación para tomar notas. Deberás obtener información del usuario, trabajar con JSON y crear un lugar para almacenar los datos del usuario.

Obtener información es esencial para crear aplicaciones del mundo real. En esta práctica deberás configurar argumentos de línea de comando que permitan a los usuarios pasar datos a tu aplicación.

Los argumentos de la línea de comandos son valores que se pasan a la aplicación desde la terminal. Una aplicación Node.js puede acceder a los argumentos de la línea de comandos que se proporcionaron mediante process.argv. Esta matriz contiene al menos dos elementos. El primero es la ruta al ejecutable de Node.js. El segundo es la ruta al archivo JavaScript que se ejecutó. Todo lo que sigue es un argumento de línea de comandos.

```
const command = process.argv[2]
if (command === 'add') {
console.log('Adding note!')
} else if (command === 'remove') {
console.log('Removing note!')
}
```

Node.js proporciona una forma básica de acceder a los argumentos de la línea de comandos. Si bien es un buen comienzo, no proporciona ninguna forma de analizar argumentos de línea de comandos más complejos. Utiliza el paquete Yargs para configurar fácilmente un conjunto de argumentos más complejo para tu aplicación.

Primero, instala Yargs en tu proyecto.

npm i yargs@17.2.1

El siguiente ejemplo muestra cómo se puede usar yargs. Primero, yargs.version se usa para configurar una versión para la herramienta de línea de comando. A continuación, yargs.command se usa para agregar soporte para un nuevo comando.

```
const yargs = require('yargs')
yargs.version('1.1.0')
yargs.command({
  command: 'add',
```

```
  describe: 'Add a new note',
  handler: function () {
  console.log('Adding a new note!')
  }
})
console.log(yargs.argv)
```

El objetivo es permitir que los usuarios pasen el título y el cuerpo de sus notas utilizando las opciones de la línea de comandos. Esta misma técnica podría usarse para permitir que los usuarios pasen datos como su nombre, correo electrónico o dirección.

Las opciones son información adicional que se transmite junto con el comando. Puedes configurar opciones para un comando utilizando la propiedad del constructor como se muestra a continuación.

El comando *add* se puede usar con dos opciones. El primero es el título que se usa para el título de la nota que se agrega. El segundo es el cuerpo que se usa para el cuerpo de la nota que se agrega. Ambas opciones son necesarias porque *demandOption* se establece en true. Ambos también están configurados para aceptar la entrada de cadena porque el tipo se establece en 'string'.

```
yargs.command({
    command: 'add',
    describe: 'Add a new note',
    builder: {
    title: {
    describe: 'Note title',
    demandOption: true,
    type: 'string'
    },
    body: {
    describe: 'Note body',
    demandOption: true,
    type: 'string'
    }
    },
    handler: function (argv) {
    console.log('Title: ' + argv.title)
    console.log('Body: ' + argv.body)
    }
})
```

El comando add ahora puede usarse con –title y --body

```
$ node app.js add --title="Buy" --body="Note body here"
Title: Buy
Body: Note body here
```

Trabajando con JSON

Dado que JSON no es más que una cadena, se puede usar para almacenar datos en un archivo de texto o transferir datos a través de solicitudes HTTP entre dos máquinas.

JavaScript proporciona dos métodos para trabajar con JSON. El primero es JSON.stringify y el segundo es JSON.parse. JSON.stringify convierte un objeto JavaScript en una cadena JSON, mientras que JSON.parse convierte una cadena JSON en un objeto JavaScript.

```
const book = {
title: 'Ego is the Enemy',
author: 'Ryan Holiday'
}
// Covert JavaScript object into JSON string
const bookJSON = JSON.stringify(book)
// Covert JSON string into object
const bookObject = JSON.parse(bookJSON)
console.log(bookObject.title) // Print: Ego is the Enemy
```

JSON se parece a un objeto de JavaScript, pero existen algunas diferencias. La más obvia es que todas las propiedades están entre comillas dobles. Las comillas simples no se pueden usar aquí, ya que JSON solo admite comillas dobles.

```
{"name":"Gunther","planet":"Earth","age":54}
```

Tu aplicación debe tener la funcionalidad de agregar, eliminar y consultar notas. Como apoyo puedes revisar el código de esta aplicación en el siguiente enlace:

https://github.com/andrewjmead/node-course-v3-code/tree/master/notes-app

Comparte el enlace de tu repositorio GitHub donde subiste tu aplicación.