**IT 기술 실무**

**기말고사 #2**

이름 : 김민제

학번 : 202312719

Repository 주소 : <https://github.com/kimminje1015/-IT-/tree/main/Lecture_2023-1-main/%EA%B8%B0%EB%A7%90%EA%B3%A0%EC%82%AC%232>

CLI를 이용해서 오늘의 할일 목록 관리 프로그램을 만들었습니다.

소스 설명

파일은 package.json, nodeCli.js 오늘의 할일(Date).txt 로 구성 되어 있습니다. 하지만txt파일은 오늘의 할일 만 관리하는 프로그램이랑 하루가 지나면 새로운 Date의 txt가 생성된다.

package.json

  "bin": {

    "cli": "./cli.js"

  }

bin : { “cli”: “./cli.js”} 는 콘솔 명령어를 cli라는 명령어로 하겠다는 뜻입니다. 실행하면 해당 cli.js가 실행됩니다.

nodeCli.js

#!/usr/bin/env node

//node 경로 맥or리눅스

해당 소스는 주석 처리된 소스다 하지만 윈도우에서는 주석 처리가 되지만 맥이나 리눅스에서는 node의 경로를 알려주는 역할을 한다.

const fs = require('fs');

const readline = require('readline');

fs 모듈과 readline 모듈은 npm 내장 모듈이다 그래서 package-lock.json 파일이랑 node\_modules폴더가 없는 것이다. 굳이 설치를 안 해도 처음에 npm init를 할 때 같이 설치되는 내장 모듈이다. 해당 소스를 모듈을 변수화한 것이다.

fs모듈은 파일을 읽고 쓰는 등의 파일 시스템 작업을 수행할 수 있는 기능을 제공한합니다.

readline 모듈은 터미널에서 사용자 입력을 읽거나 터미널에 출력을 작성하는 기능을 제공합니다.

//stdin 터미널에서 입력

//stdout 터미널에서 결과 출력

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout,

});

readline.createInterface()함수는 사용해서 readline.Interface 객체를 생성합니다. createInterface()함수는 설정 객체를 전달하는데 설정 객체는 input, 과 output이 있습니다. input은 표준 입력 스트림으로 사용할 스트림 객체를 설정합니다. 일반적으로 process.stdin을 사용하여 터미널의 표준 입력을 읽습니다. output은 표준 출력 스트림으로 사용할 스트림 객체를 설정합니다. 일반적으로 process.stdout을 사용해서 터미널의 표준 출력에 내용을 출력합니다.

//오늘 할 목록 추가

function addTodayItem(todayItem) {

  fs.readFile(todayListFileName, 'utf8', (err, data) => {

    if(data === '')

    {

      fs.appendFile(todayListFileName, `${todayItem}`, (err) => {

        if (err) {

          console.error('오늘의 할일을 추가하는 도중 오류가 발생했습니다.', err);

          console.log(`===================================`);

        }

        else {

          console.log(`[${todayItem}] 오늘의 할일이 추가되었습니다.`);

          console.log(`===================================`);

        }

        displayMenu();

        askUser();

      });

    }

    else

    {

      fs.appendFile(todayListFileName, `\n${todayItem}`, (err) => {

        if (err) {

          console.error('오늘의 할일을 추가하는 도중 오류가 발생했습니다.', err);

          console.log(`===================================`);

        }

        else {

          console.log(`[${todayItem}] 오늘의 할일이 추가되었습니다.`);

          console.log(`===================================`);

        }

        displayMenu();

        askUser();

      });

    }

  });

}

fs.readFile은 파일을 읽어 오는 함수다. 읽은 파일은 내용(data)이 null이면 작성한 내용(todayItem)을 파일에 추가하고 읽은 파일의 내용(data)이 이미 있으면 작성한 내용(todayItem)으로 파일에 추가하되 \n를 이용해서 띄어쓰기 후 작성한 내용(todayItem)을 추가한다.

fs.appendFile은 지정한 파일에 데이터를 추가하는 함수다.

//삭제 할 목록 보기

function deletetDisplayTodayList() {

  fs.readFile(todayListFileName, 'utf8', (err, data) => {

    if (err) {

      console.error('오늘의 할일 목록을 읽어오는 도중 오류가 발생했습니다.', err);

      console.log(`===================================`);

    } else {

      if (data.trim() === '') {

        console.log('삭제할 오늘의 할일이 없습니다.');

        console.log(`===================================`);

        displayMenu();

        askUser();

      } else {

        const todayList = data.split('\n');

        console.log('===== 오늘의 할일 목록 =====');

        todayList.forEach((item, index) => {

          console.log(`${index + 1}. ${item}`);

        });

        rl.question('삭제할 오늘의 할일 번호를 입력하세요: ', (index) => {

          deletetodayItem(parseInt(index));

        });

      }

    }

  });

}

rl.question() 함수는 readline 모듈의 메서드로 사용자로부터 콘솔로 질문을 표시하고 사용자의 입력을 받을 수 있는 함수다. 해당 함수를 사용해서 콘솔에서 질문과 동시에 삭제할 오늘의 할일 번호 또한 사용자한테 입력 받습니다. 비동기적으로 작동하기 때문에 콜백 함수를 통해서 사용자의 입력 처리하고 추가적인 작업 처리가 가능합니다.

//목록 삭제

function deletetodayItem(index) {

  fs.readFile(todayListFileName, 'utf8', (err, data) => {

    if (err) {

      console.error('오늘의 할일 목록을 읽어오는 도중 오류가 발생했습니다.', err);

      console.log(`===================================`);

    } else {

      const todayList = data.split('\n');

      if (index >= 1 && index <= todayList.length) {

        const deletedItem = todayList.splice(index - 1, 1);

        fs.writeFile(todayListFileName, todayList.join('\n'), (err) => {

          if (err) {

            console.error('할일을 삭제하는 도중 오류가 발생했습니다.', err);

            console.log(`===================================`);

          } else {

            console.log(`[${deletedItem}] 할일이 삭제되었습니다.`);

            console.log(`===================================`);

          }

          displayMenu();

          askUser();

        });

      } else {

        console.log('올바른 번호를 입력하세요.');

        displayMenu();

        askUser();

      }

    }

  });

}

fs.writeFile() 함수는 파일에 데이터를 추가하는 함수다. 위에서 fs.appendFile()함수랑 비슷하다고 생각하겠지만 fs.writeFile()함수는 파일에 데이터가 이미 있으면 이전 데이터가 덮어쓰여집니다. fs.appendFile()함수는 파일의 데이터 끝에 추가하는 함수라는 차이점이 있습니다.

//콘솔 지우기

function clearDisplayTodayList(){

  console.clear();

  displayMenu();

  askUser();

}

콘솔의 내용을 지우는 역할을 합니다 CLI(Command-Line Interface)은 명령어 입력으로 지시를 하는 특징이 있어서 명령어들이 많아지면 가독성이 떨어지는 불편함 때문에 사용했습니다.

// 메인 로직

fs.access(todayListFileName, fs.constants.F\_OK, (err) => {

  if (err) {

    // 오늘의 할일(Date).txt 파일이 없는 경우 파일을 생성합니다.

    fs.writeFile(todayListFileName, '', (err) => {

      if (err) {

        console.error('오늘의 할일 목록 파일을 생성하는 도중 오류가 발생했습니다.', err);

      }

      displayMenu();

      askUser();

    });

  } else {

    displayMenu();

    askUser();

  }

});

fs.access() 함수는 파일의 존재 여부를 확인하고 파일이 존재하지 않는 경우 파일을 생성하는 함수다.

실행 결과

