**IT 기술 실무**

**기말고사 #1**

이름 : 김민제

학번 : 202312719

Repository 주소 : <https://github.com/kimminje1015/-IT-/tree/main/Lecture_2023-1-main/기말고사#1>

Web API 서버랑 Web Socket 을 이용해서 실시간 주식(난수)프로그램(실제로 주식 정보를 가져오지 못해서 해당 프로그램은 난수를 이용한 프로그램입니다)을 만들었습니다.

소스 설명

파일은 package-lock.json, package.json, Stock.html, Stock.js 으로 나눠져 있습니다.

package.json

{

  "name": "node",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "author": "",

  "license": "ISC",

  "dependencies": {

    "cors": "^2.8.5",

    "express": "^4.18.2",

    "http": "^0.0.1-security",

    "ws": "^8.13.0"

  }

}

npm은 CORS, express, http, ws들의 모듈들을 사용했습니다. 해당 모듈은 npm를 통해서 설치를 하면 자동으로 소스가 설정된다.

CORS은 Cross-Origin Resource Sharing (CORS)를 처리하기 위한 패키지입니다.

다른 도메인의 요청을 허용하기 위해서 사용했습니다. 하지만 여기서는 fetch나 axios를 사용하지 않아서 불필요하긴 하지만 조금더 보안해서 fetch나 axios등 다른 도메인을 이용할 때 사용할라고 설치했습니다.

Express는 Node.js를 위한 웹 애플리케이션 프레임워크입니다.

미들웨어와 라우팅 기능을 제공하여 웹 애플리케이션의 개발을 간편하게 할 수 있습니다.

Node.js의 내장 모듈인 "http"는 HTTP 프로토콜을 사용하는 웹 서버를 만들기 위해 사용됩니다.웹 서버를 만들고 HTTP 요청 및 응답을 처리하는 데 필요한 핵심 기능을 제공합니다. http" 모듈을 사용하면 웹 애플리케이션을 개발할 수 있지만, 보편적으로 Express.js와 같은 외부 모듈과 함께 사용됩니다.

ws(WebSocket) 프로토콜을 사용하여 양방향 통신을 제공하는 패키지입니다.

클라이언트와 서버 간 실시간 양방향 데이터 전송을 위한 웹 소켓 서버와 클라이언트를 개발할 수 있습니다. "ws" 패키지를 사용하면 서버와 클라이언트 간의 지속적인 연결을 유지하고, 실시간 데이터를 주고받을 수 있습니다.

지금 까지 설치한 npm 에 대한 설명이였습니다. Web API서버를 만들기 위해서 Express를 사용했고 Web Socket를 만들기 위해서 ws를 사용했습니다.

package-lock.json

"node\_modules/accepts": {

      "version": "1.3.8",

      "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.8.tgz",

      "integrity": "sha512-PYAthTa2m2VKxuvSD3DPC/Gy+U+sOA1LAuT8mkmRuvw+NACSaeXEQ+NHcVF7rONl6qcaxV3Uuemwawk+7+SJLw==",

      "dependencies": {

        "mime-types": "~2.1.34",

        "negotiator": "0.6.3"

      },

      "engines": {

        "node": ">= 0.6"

      }

    },

    "node\_modules/array-flatten": {

      "version": "1.1.1",

package-lock.json은 package.json를 세부화한 소스다 npm설치를 하면 자동으로 생성된다.

Stock.js

const express = require('express');

const http = require('http');

const WebSocket = require('ws');

const cors = require('cors');

해당 소스는 npm들을 변수화 한 소스입니다.

해당 모듈을 가져올때는 require()라는 함수를 사용한다. 그래서 위해서 npm를 설명한 express, http, ws(WebSocket), cors(cors)를 모듈을 해서 가졌왔다.

// CORS 미들웨어 사용

// app.use(cors());

해당 소스는 CORS를 사용해겠다는 뜻인데 위에서 설명하듯이 다른 도메인을 사용하지 않아서 주석처리 했습니다.

const app = express();

const server = http.createServer(app);

const wss = new WebSocket.Server({ server });

http.createServer를 사용해서 HTTP서버를 생성하고, 해당 서버를 기반으로 Socket서버를 설정해습니다. express() 함수를 변수화한 app를 기반으로 HTTP 서버를 생성했으니 이를 통해 웹 서버를 동작할 수 있다.

// API 엔드포인트 - 현재 주식 시세

app.get('/', (req, res) => {

  res.sendFile(\_\_dirname + '/Stock.html');

});

해당 소스는 app.get(라우터, (req,res) =>{})는 / 해당 라우터 에서(/ 이니깐 localhost:3333/) (req,res) =>{} GET이 요청한 값을 보여주겠다는 뜻입니다.

res.sendFile(\_\_dirname + '/Stock.html')에서 res.sendFile() 메서드를 사용하여 Stock.html 파일을 클라이언트에게 전송합니다. \_\_dirname은 현재 파일이 위치한 디렉토리 경로를 나타내며, 해당 파일의 디렉토리에 있는 Stock.html 파일을 클라이언트에게 전송합니다.

// WebSocket 연결 시 발생하는 이벤트 처리

**wss.on('connection', (ws) => {**

**console.log('Client connected');**

**// 0.3초마다 주식 시세 업데이트 및 클라이언트에 전송**

**const interval = setInterval(() => {**

**const stock = generateRandomStockData();**

**ws.send(JSON.stringify(stock));**

**}, 300);**

**// 클라이언트와의 연결이 종료됐을 때 처리**

**ws.on('close', () => {**

**console.log('Client disconnected');**

**clearInterval(interval);**

**});**

**});**

WebSocket 서버(wss)에서 클라이언트와의 연결이 수립되면 호출되는 이벤트 핸들러입니다.

setInterval()은 클라이언트에 전송하는 함수입니다. 여기서 300으로 설정을 하면 0.3초 마다 해당 클라이언트에 전송합니다.

generateRandomStockData()은 밑에 설명한 난수를 발생시키는 함수고 ws.send은 generateRandomStockData()에서 나온 난수를 클라이언트에 보내준다 보내주는 과정에서 JOSN.stringift를 사용해서 JOSN 형식으로 보내진다.

clearInterval(interval)은 연결이 종료되면 이전에 설정한 타이머(setInterval(,300))를 해제하는 함수다.

// 랜덤 주식 생성

function generateRandomStockData() {

  const symbols = ['애플', '삼성', '테슬라','LG','블리자드','넥슨','다이소','하이닉스','MSI','인텔','AMD'];

  const randomSymbol = symbols[Math.floor(Math.random() \* symbols.length)];

  const price = Math.random() \* 1000;

  return { symbol: randomSymbol, price: price.toFixed(2) };

}

해당 소스는 주식(난수)를 생성하는 소스다. 주식API를 가져와서 적용하면 실시간 주식를 보여주는 프로그램이 되지만 주식API를 사용하지 못해서 난수를 사용했습니다.

// 서버 시작 포트 3333

const port = 3333;

server.listen(port, () => {

  console.log(`Server listening on port ${port}`);

});

server.listen은 해당 포트에 바인딩 해주는 함수다. port는 3333으로 변수를 지정했으니깐 localhost:3333으로 포트를 바인딩해준다. 해당 포트가 아무 번호를 사용해도 무관한다.

그리고 포트 다음에 오는 콜백함수는 그 포트(3333)에 연결하면 호출되는 함수다 콜백함수는 없어도 무관하다.

Stock.html

    //포트 연결

    const socket = new WebSocket('ws://localhost:3333');

    const stockTable = document.getElementById('stockTable').getElementsByTagName('tbody')[0];

    socket.onmessage = (event) => {

      const stock = JSON.parse(event.data);

      updateStockTable(stock);

    };

html은 위 소스 말고는 html 소스라서 설명을 생략하겠습니다.

new WebSocket은 객체(WebSocket)를 생성해서 socket이라는 변수에 넣는다. 이때 해당 포트는 Stock.js에서 설명한 3333포트로 연결했습니다.

.onmessage는 서버로부터 수신할 때마다 호출 되는 함수다. 이 함수를 통해서 서버에 수신할때마다(0.3초 간격) 새로운 난수를 보여준다.

실행 결과

