TOC

Node.js 란

- V8
  - Chrome V8 Javascript 엔진으로 빌드된 Javascript 런타임
- Event Loop
  - 콜스택 -> 콜백큐
- Non-Blocking I/O
  - = 비동기식 I/O
- 싱글 스레드
  - 하나의 마스터 프로세스 -> CPU 개수만큼 워커 프로세스를 생성

개발환경 구성

- Visual Studio Code 설치
- Node.js 설치
  - 20.13.0(LTS) 다운로드 https://nodejs.org/en/download
  - 설치 후 확인

```
$ node -v                    ==> v20.13.0
$ npm -v                     ==> 8.18.0
```

- VS Code 터미널 모드 변경

F1

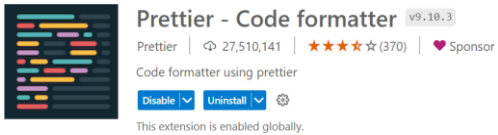
View ->
Commend Palette

개발환경 구성

- VS code Extension 설치
  - JavaScript (ES6) snippets
    - 자바스크립트 코드 자동완성

  - ESLint
    - 자바스크립트 문법 오류 체크

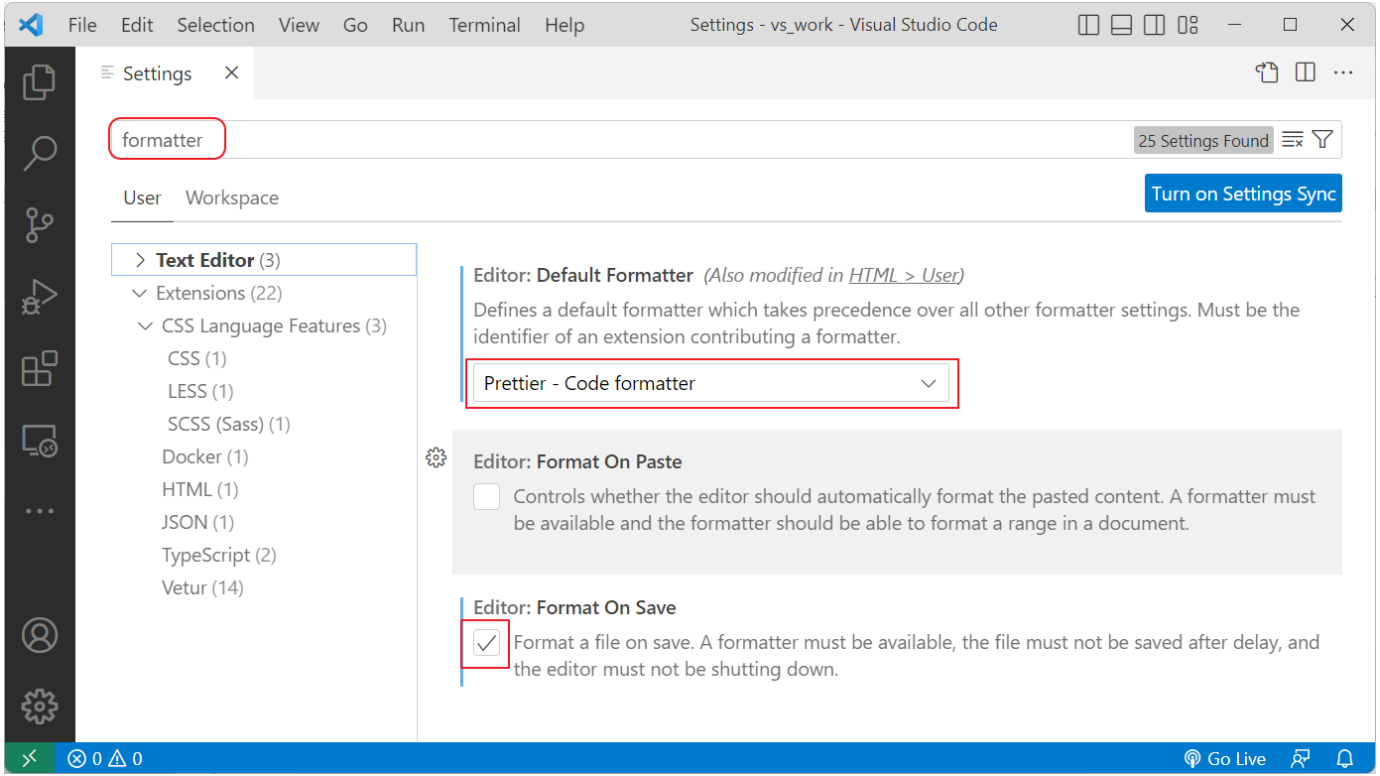
  - Pretter - Code Formatter
    - 미리 지정된 코드포맷 스타일로 자동변경



  - Live Server

개발환경 구성

- VS code setting
  - File -> Preferences -> Settings `Ctrl + ,`



Node.js 시작하기

- **자바스크립트 실행환경**

```
> node
```

```
> node
Welcome to Node.js v20.11.0.
Type ".help" for more information.
> console.log("node~~~")
node~~~
```

- **스크립트 파일 실행**

```
console.log("node~~~")
```

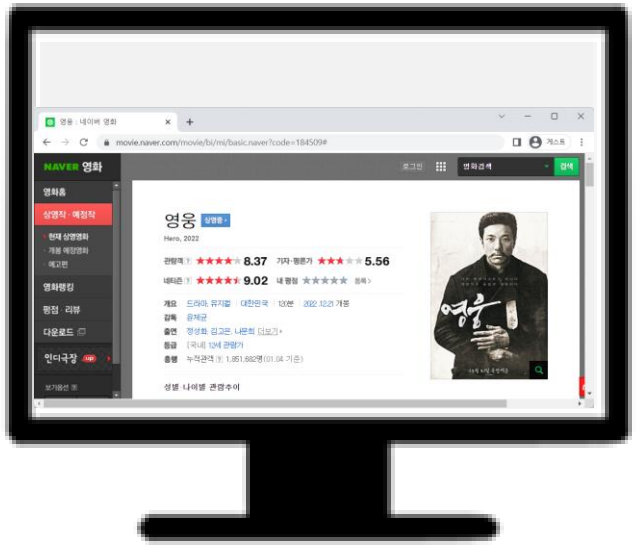
test.js 파일 생성

```
> node test.js
```

```
> node test.js
node~~~
```

웹서버

http://도메인:포트/경로/영웅.html



영웅.html, 아바타.html , …

```
<head>
<style>
  .movieNm {}
</style>
</head>
<body>
  <h1 class="movieNm">영웅</h1>
  <h3>기본정보</h3>
  <div>
    <div>개요
      <span class="genreNm">드라마</span>
      <span class="nationNm">대한민국</span>
      <span class="showTm">120분</span>
    </div>
    <div>개봉<span class="openDt">2022.12.21</span></div>
    <p class="info">어머니 조마리아와.....</p>
  </div>
  <div><img src="" class="poster"></div>
</body>
```

http://도메인/movie.html/영웅

movie.html (뷰 템플릿)

```
<body>
  <h1 class="movieNm">영화제목</h1>
  <h3>기본정보</h3>
  <div>
    <div>개요
      <span class="genreNm">장르</span>
      <span class="nationNm">제작국가</span>
…
<script>
fecth("http://도메인/movie/영웅")
  .then(res=>res.json())
  .then(res=>{
    //서버에서 영화정보를 요청
    //응답결과를 각각의 태그위치에 출력
    document.querySelector(".genreNm")
            .innerHTML = res.genreNm
})
</script>
</body>
```

WAS(rest 서버)

```
router.get("/movie/:id", (req, res) => {
  const id = req.params.id;
  sql = "SELECT * FROM movies where movieCd=?";
  pool.query(sql, id, function (err, results, fields) {
    res.json(results);
  });
});
```

영화정보 데이터베이스

movieCd	movieNm	nationNm	genreNm	info
2022122	영웅	대한민국	드라마	
2022121	아바타	미국	SF	

movie/2022122


```
router.get("/:id", (req, res) => {
  const id = req.params.id;
  sql = "SELECT * FROM movies where movieCd=?";
  pool.query(sql, id, function (err, results, fields) {
      res.render("movie", results);
  });
});
```

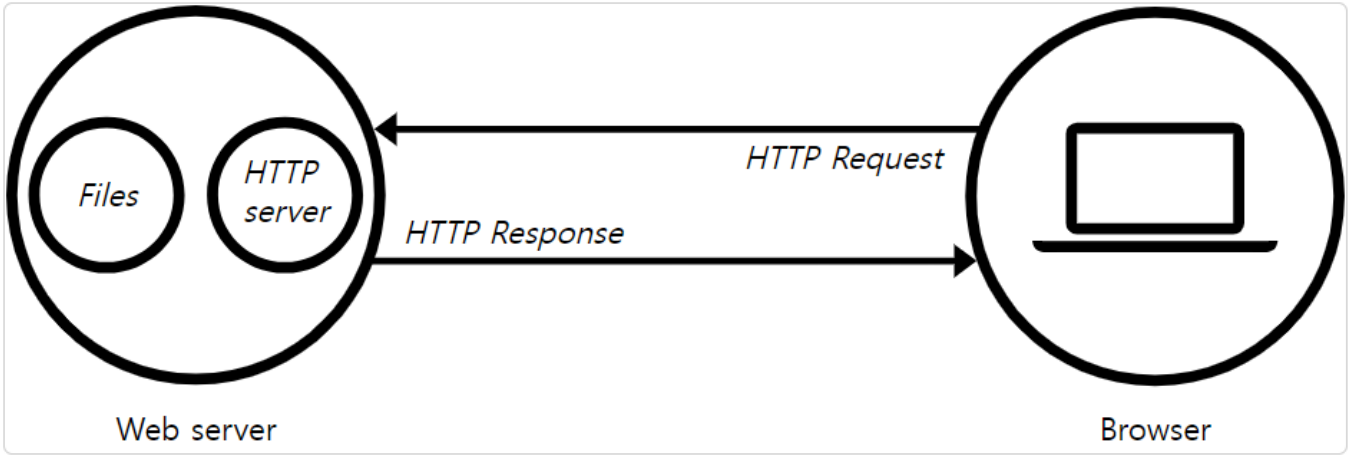
movie.ejs (뷰 템플릿)

```
<body>
  <h1 class="movieNm">영웅</h1>
  <h3>기본정보</h3>
  <div>
    <div>개요
      <span class="genreNm"><%=movie.genreNm%></span>
      <span class="showTm"><%=movie.showTm%></span>
      <span class="openDt"><%=movie.openDt%></span>
    </div>
    <p class="info"><%=movie.info%></p>
  </div>
  <div><img src="<%=movie.poster%>" class="poster"></div>
</body>
```

영화정보 데이터베이스

movieCd	movieNm	genreNm
2022122	영웅	드라마
2022121	아바타	SF

- 웹서버란


Files
HTTP server
HTTP Request
HTTP Response
Web server
Browser

웹서버 만들기

- node.js 는 웹서버 기능을 내장
- http 모듈 https://nodejs.org/api/http.html

```
// 1. http 모듈 포함
const http = require("http");

// 2. http 서버 생성
// 5. 새로운 요청이 수신되면 request 이벤트가 호출되고
    /* http.IncomingMessage 객체와 http.ServerResPonse 객체를 넘겨준다
       req : 요청 상세정보( 요청 헤더와 요청 data )
       res : 클라이언트에게 데이터를 반환(상태코드, contentType, 응답 데이터 ) */
const server = http.createServer((req , res ) => {});

// 3. 지정된 포트 및 호스트이름으로 수신 대기
// 4. 서버가 준비되면 콜백함수 호출
server .listen (3000 , "127.0.0.1", () => {});
```

웹서버 만들기

```
// 1. http 모듈 포함
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;

// 2. http 서버 생성(요청이 수신되면 응답 처리)
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

// 3. 지정된 포트 및 호스트이름으로 수신 대기
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

```
D:\node_work> node app.js
Server running at http://127.0.0.1:3000/
```

- http 응답

```
const server = http.createServer(function (req, res) {
    res.statusCode = 200;
    res.setHeader("Content-Type", "text/html");
    res.writeHead(200, { 'Content-Type':"text/html" });
    res.write(`<html lang='ko'><head></head><body>node 서버</body></html>`);
    res.end(template);
})
```

  - StatusCode(응답코드)
    - 200(OK), 301/302(redirect), 403(forbidden), 404(not found), 500(internal server error)
  - setHeader
    - MIME 타입 :웹브라우저에게 문서를 어떻게 출력할지 결정하는 부분
  - write()
    - 클라이언트에 실제로 보내고자 하는 데이터를 전송. 여러 번 사용 가능
  - end()
    - 해당 데이터를 보낸 뒤 서버와 클라이언트와의 접속이 종료

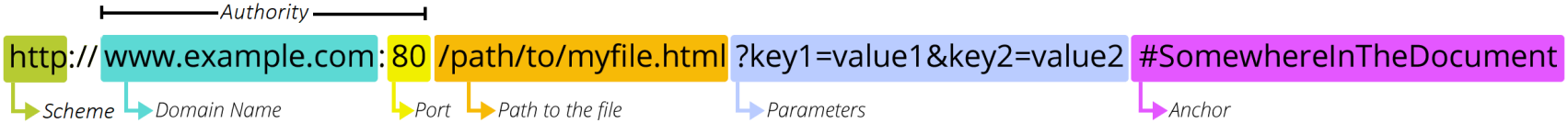
- json 응답

```
const http = require("http");

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "application/json" });
  res.end(
    JSON.stringify({
      data: "Hello World!!",
    })
  );
});

server.listen(8000);
```

URL 이해하기



- scheme(protocol)
  - 통신규칙 : ftp, https
- domain name(host)
  - 인터넷에 연결되어 있는 컴퓨터의 주소
- port
  - 한 대의 컴퓨터에 여러 개의 서버가 있는 경우 서버를 구분
- path
  - 경로
- parameter(query string)
  - 서버로 전달되는 데이터. ?name=value&name=value로 구성

참고사이트 : https://developer.mozilla.org/ko/docs/Learn/Common_questions/What_is_a_URL
https://nodejs.org/api/url.html

URL에서 Query String 추출

- URL 객체 => searchParams https://nodejs.org/api/url.html

```
const server = http.createServer(function (request, response) {
  const myURL = new URL("http://127.0.0.1" + request.url);
  console.log("pathname", myURL.pathname);
  console.log("search", myURL.searchParams);
  console.log("id", myURL.searchParams.get("id"));
  response.end("hello");
});
```

- url 모듈 => query https://www.npmjs.com/package/url

```
const url = require("url");
const server = http.createServer(function (request, response) {
  const _url = url.parse(req.url , true );
  console.log("pathname", _url.pathname);
  console.log("search", _url.query );
  console.log("id", _url.query.userid );
  response.end("hello");
});
```

```
const http = require("http");
const server = http.createServer(function (req, res) {
  const myURL = new URL("http://127.0.0.1:3000" + req.url);
  let pathname = myURL.pathname;
  if (pathname == "/") {
    res.statusCode = 200;
    res.setHeader("Content-Type", "text/plain");
    res.end("hello");
  } else if (pathname == "/info") {
    res.statusCode = 200;
    res.setHeader("Content-Type", "text/html");
    let template = `<!DOCTYPE html><html lang='ko'> <head><meta charset="UTF-8"></head>
    <body><h1 style='color:blue'>node 서버</h1></body></html>`;
    res.end(template);
  } else {
    res.statusCode = 404;
    res.end("error");
  }
}).listen(3000, function() {  console.log("server runtime http://localhost:3000"); });
```

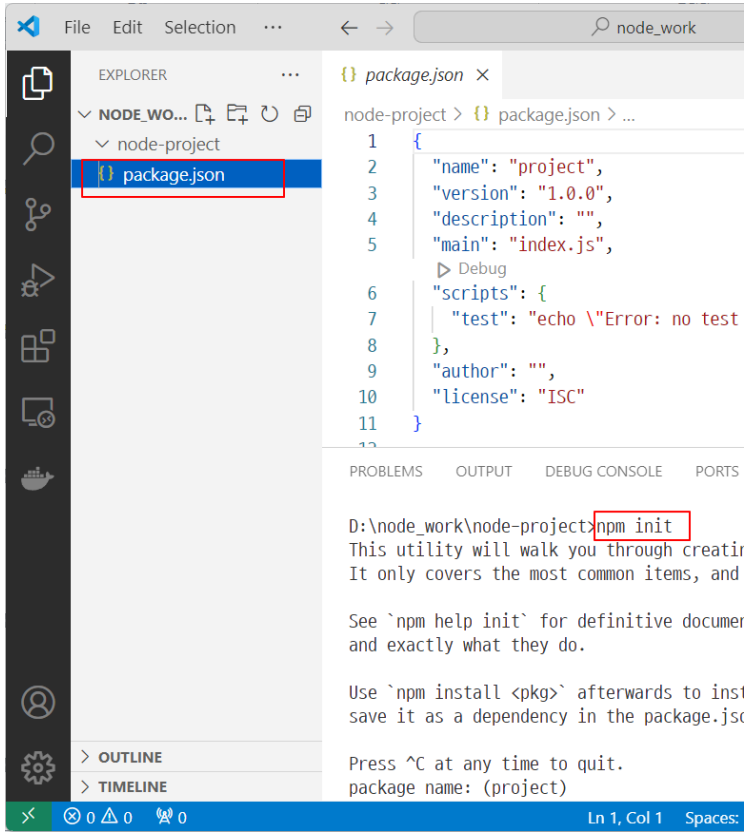
- Node Package Manager
  - 대부분의 자바스크립트 프로그램은 패키지라는 이름으로 npm 서버에 등록되어 있으므로 찾아서 설치
  - https://www.npmjs.com/
- package.json 파일
  - 설치한 패키지의 버전을 관리하고 패키지간의 의존성 관리하는 파일
  - 의존성 선언에는 version range가 사용(특정 버전이 아니라 버전의 범위)
- package-lock.json
  - 의존성 트리에 대한 정확한 버전 정보
  - node_modules 나 package.json이 수정되는 경우 생성되거나 업데이트
- node_modules 폴더
  - 다운로드 받아서 설치된 모듈 파일 위치

- package.json 파일 생성

```
> npm init
```

```
> npm init
save it as a dependency in the
package.json file.

package name:      - 프로젝트 혹은 애플리케이션 대표 이름
version:           - 패키지 버전
description:       - 패키지에 대한 설명
entry point:       - 프로젝트에서 가장 먼저 실행되는 스크립트 파일
test command:      - 코드를 테스트할 때 입력할 명령어
git repository:    - Git 저장소 주소
keywords:          - npm 검색을 위해서 제공되는 키워드
author:            - 개발자
license:           - 패키지 라이선스 기본값은 ISC(프리웨어)
```



npm 명령어

- **패키지 설치**

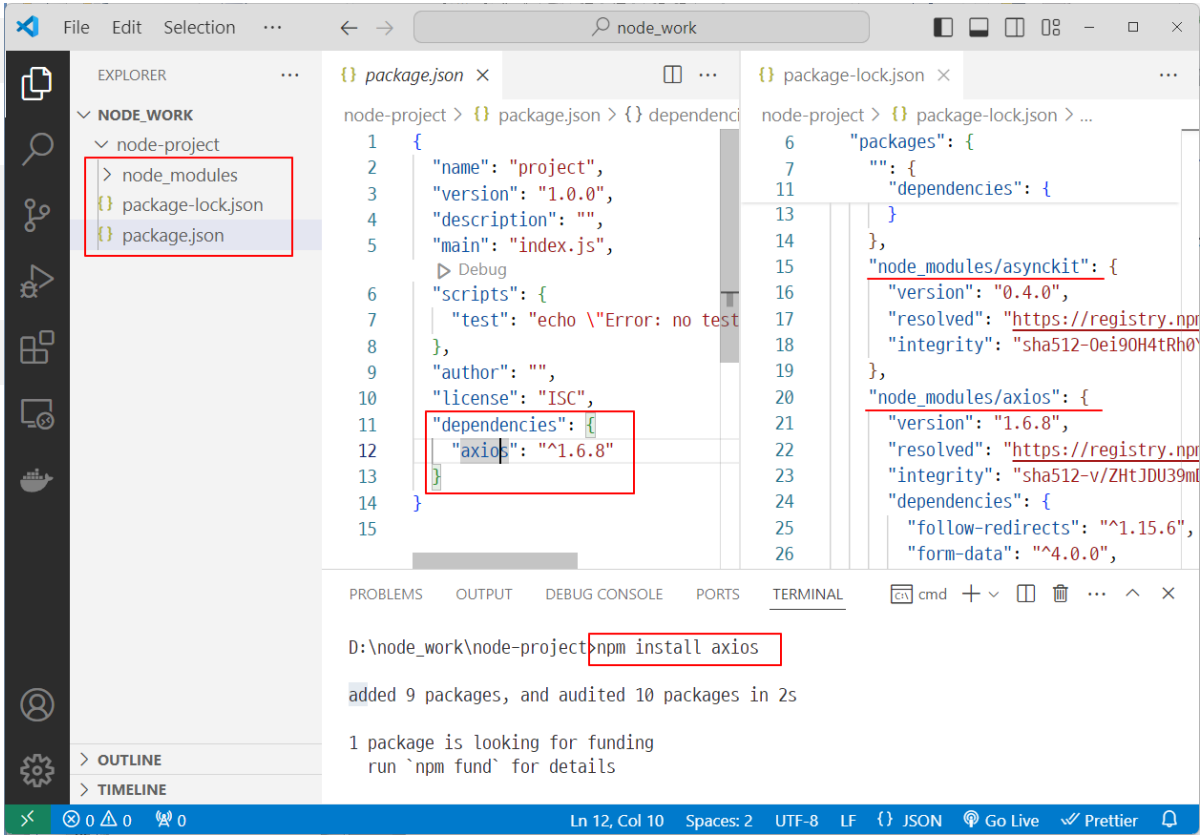
```
> npm install axios
```

  - **설치버전 지정**

```
> npm install mysql@2.11.0
```

  - **최신버전**

```
> npm install mysql@latest
```



npm 명령어

- 패키지 제거

```
> npm uninstall mysql
```

- package.json에 있는 모든 의존성이 있는 모듈을 자동으로 설치

```
> npm install
> npm update
```

- 패키지 업데이트 확인

```
> npm outdated
```

npm 명령어

- node_modloes의 위치 확인

```
npm root
```

```
> npm root
D:\node_work\npmtest\node_modules
```

- 패키지 조회

```
npm ls
```

```
D:\node_work\npmtest>npm ls
node_week1@ D:\node_work\npmtest
├── global@4.4.0
└── pm2@5.2.2
```

- yarn
  - npm 대체자로 페이스북이 내놓은 패키지 매니저
  - 따로 설치해야 하며 npm이 느릴 경우 yarn 패키지로 대신 설치 가능
  - npm으로 설치

```
> npm install –g yarn
```

  - choco 로 설치

```
> choco install yarn
```

  - yarn 명령어

```
> yarn init              // yarn.lock 파일이 생성됨
> yarn install
> yarn add 패키지명
> yarn upgrade [패키지명]
> yarn remove 패키지명
```

CommonJS modules

- module.exports
- require()

```
const axios = require("axios");

const reqUrl  = "https://jsonplaceholder.typicode.com/todos/1"
axios(reqUrl). then( (res)  =>  console.log(res.data)  );
```

ES modules

- export
- import

- Process Manager
  - node.js 어플리케이션 프로세스 메니저 https://pm2.keymetrics.io/
  - 무중단 서비스 지원 -가동 중지시간없이 응용프로그램을 영원히 활성상태로 유지(자동으로 리로드)
  - 현재 디렉터리 또는 하위 디렉터리에서 파일이 수정될 때 애플리케이션을 자동으로 다시 시작.
- npm 설치

```
npm install pm2 -g
```

- application 실행

```
pm2 start app.js [--watch]
```

- application 관리

```
pm2 list
pm2 stop    app_name
pm2 restart app_name
pm2 delete  app_name
```

- log 보기

```
pm2 logs [app-name]
pm2 logs --time
pm2 logs --json
```

- 실행중인 모든 프로세스 모니터링

```
 pm2 monit
```

nodemon

- Node.js 애플리케이션을 개발하는 동안 변경사항을 감지하고 자동으로 서버를 재시작해주는 도구

- nodemon 설치

```
> npm install -g nodemon
```

- application 실행
  - src 디렉토리에서 코드변화가 감지되면 src/index.js를 재시작

```
> nodemon --watch src/ src/index.js
```

  - 애플리케이션 변경사항 모니터링

```
> nodemon app.js
```

json-server

- JSON 기반으로 가상의 REST API 서버를 구축할 수 있는 npm 모듈
- json-server 설치

```
> npm install –g json-server
```

- json 파일 생성

db.json

```
{
  "posts" :  [ {"id": 1,"title": "json-server", "author": "hong"} ] ,
  "comments" :  [ {"id": 1, "body": "some comment", "postId":1} ] ,
  " profile" :  [ {"name":  "hong"} ] ,
}
```

- json-server 실행

```
> json-server –watch db.json
```

GET 요청방식

▶ queryString

```
<form method="/movie" action="get">
  <input name="movieCd" value="20221201">
  <button>조회</button>
</form>
```

```
<a href="/movie?movieCd=20221201">상세정보</a>
```

```
location.href="/movie?movieCd=20221201“;
```

```
fecth("/movie?movieCd=20221201")
```

▶ path Variable

```
<a href="/movie/20221201">상세정보</a>
```

```
location.href="/movie/20221201";
```

```
fecth("/movie/20221201")
```

POST 요청방식

▶ queryString

```
<form method="/movie" action="post">
  <input name="movieCd" value="20221201">
  <button>조회</button>
</form>
```

```
fecth("/movie", {
   method:"post“,
   headers: {'Content-Type':
             'application/x-www-form-urlencoded'},
   body: "movieCd=20221201"
}).then(res=>res.json())
```

▶ json

```
fecth("/movie", {
   method:"post“,
   headers: {'Content-Type':'application/json' },
   body: JSON.stringify({"movieCd":"20221201"})
}).then(res=>res.json())
```