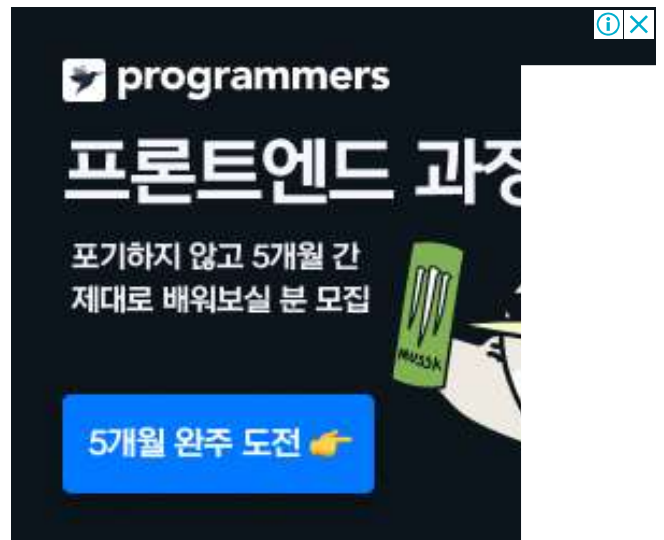
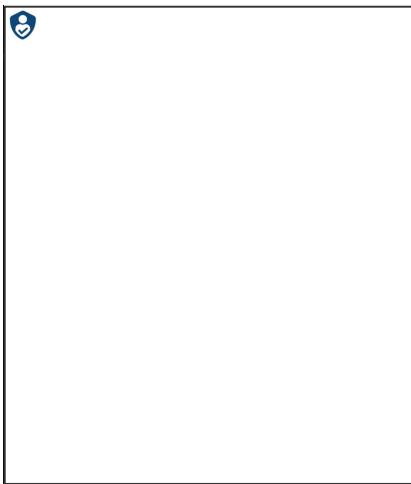


강좌4 - NodeJS - 6년 전 등록 / 3년 전 수정

Express 프레임워크

Node.js 교과서 책이 출간되었습니다. 이 포스팅보다는 책이나 동영상 강의를 보시는 것을 추천합니다.



안녕하세요 이번 시간에는 Node.js에서 제일 유명한 **Express** 프레임워크에 대해 알아보겠습니다!

Express

Express 프레임워크 외에도 **Koa**나 **Hapi**같은 **프레임워크**가 더 있는데요. 다른 두 개는 제가 안 써봐서 잘 모르겠습니다. Express는 대다수가 사용할 정도로 대중적이고 인기가 가장 많습니다.

전 시간처럼 서버의 페이지 개수가 늘어나고 사이트가 커지면 코드가 복잡해지는데요. Express 프레임워크는 코드의 양도 줄여주고 추후 유지보수가 쉽도록 만들어주기 때문에 사용합니다. 하지만 성능은 이전 시간의 생 코드보다는 떨어집니다. 하지만 크게 영향은 없기 때문에 사용해도 됩니다. 드디어 npm을 사용할 때가 왔네요. 이전의 **http**, **path**, **fs**, **url** 등의 모듈은 Node.js에서 기본 제공하는 패키지였기 때문에 설치할 필요가 없지만 Express는 Node.js가 아닌 다른 개인이나 단체가 만든 패키지이기 때문에 npm에서 다운로드받아야 합니다. cmd에서 프로젝트 폴더로 간 후

```
1 npm install express
```

로 설치해줍니다. 설치한 express는 `node_modules`라는 폴더 안에 저장됩니다. 또한 `package.json`에 express가 설치되었음이 기록됩니다. 앞으로는 `package.json` 파일만 있으면 `node_modules` 폴더가 지워졌더라도 복구할 수 있습니다.

목록



v7.3.0

로그인      회원가입

```

1 const express = require('express');
2 const app = express();
3 app.get('/', (req, res) => {
4   res.send('Hello World!');
5 });
6 app.listen(8080, () => {
7   console.log('Express App on port 8080!');
8 });

```

http 모듈은 Express에서 내부적으로 처리하기 때문에 더 이상 사용하지 않아도 됩니다. `const app = express();` 을 사용해서 만든 Express app 객체로 모든 서버의 일을 처리합니다. 위의 코드는 간단한 Hello World! 문자열을 전송합니다. 마지막으로 `app.listen` 메소드로 `http.listen` 과 같이 8080포트에서 요청을 대기하고 있죠.

단순한 문자열을 전송하는 것 대신 이전 강좌처럼 about과 main 페이지로 나누어서 응답받을 수 있게 해볼까요?

Routing

```

1 const express = require('express');
2 const path = require('path');
3 const app = express();
4 app.use(express.static(path.join(__dirname, 'html')));
5 app.get('/', (req, res) => {
6   res.sendFile(path.join(__dirname, 'html', 'main.html'));
7 });
8 app.get('/about', (req, res) => {
9   res.sendFile(path.join(__dirname, 'html', 'about.html'));
10 });
11 app.listen(8080, () => {
12   console.log('Express App on port 8080!');
13 });

```

확실히 깔끔해졌음을 느끼시나요? `server.js`를 위와 같이 바꾸고, `html` 폴더를 `server.js`와 같은 위치로 새로 만든 뒤 `main.html`, `about.html`, `turn.js`를 `html` 폴더 안으로 옮깁니다.

```

server.js
html
--main.html
--about.html
--turn.js
node_modules
package.json

```

이런 구조가 되게요. `app.use(express.static(path.join(__dirname, 'html')));` 이 코드가 추가되었는데요. `app.use`는 **미들웨어**를 쓰는 부분이고, `express.static`은 **정적 파일**들이 있는 위치를 지정하는 부분입니다. 미들웨어는 다음 강좌에서 설명하겠습니다! 지금은 그냥 정적 파일을 제공하기 위해 코드를 넣었다고만 이해하고 넘어가주세요. 정적 파일이란 나중에 배울 `ejs`나 `pug` 같은 템플릿이 아닌 그냥 파일이라고 생각하시면 됩니다. 이미지, 동영상, HTML, CSS, JS 파



목록





`path.join` 메소드를 사용하면 디렉토리 주소 구분자가 /인지 \인지 상관없이 환경에 맞게 주소를 완성해줍니다. (₩와 /는 맥, 리눅스, 윈도우 OS에 따라 다릅니다) 그리고 `..`(현재 폴더)과 `..`(상위 폴더)도 알아서 계산해서 해당 주소를 표시해주기 때문에 주소를 어떻게 써야할 지 헷갈릴 때 사용하세요.

```
1 const path = require('path');
2 path.join('example', 'children', 'file1.js'); // 'example/children/file1.js'
3 path.join('example', 'children', '..', 'file2.js'); // 'example/file2.js'
4 path.join('example', 'children', 'grandson', '..', '..', 'file3.js'); // 'example/file3.js'
```

위와 같이 `..`이 있을 경우 상위 폴더로 한 단계 올라갑니다.

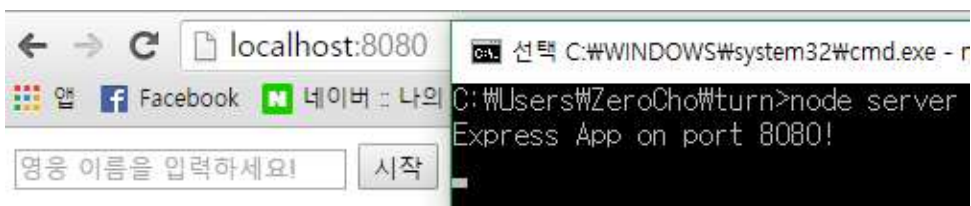
정적 파일의 위치를 `html` 폴더로 정해주면(폴더의 이름은 상관없습니다), 클라이언트에서 접근할 때 `html/[파일 이름]`으로 접근하는 게 아니라, 그냥 `[파일 이름]`으로 접근할 수 있습니다. 예를 들면 사용자가 `html` 폴더 안의 `turn.js` 파일을 보고 싶으면 주소창에 `localhost:8080/html/turn.js`를 입력하는 게 아니라, `localhost:8080/turn.js` 이렇게 입력해야 합니다. 정작 `/html/turn.js`를 하면 아무것도 접근할 수 없습니다. 사용자에게 보여주고 싶은 파일들만 `html` 폴더에 넣고, 나머지는 폴더 밖에 놓으면 **public** 폴더와 **private** 폴더를 구분할 수 있는거죠. 서버 코드같은 민감한 것들은 감출 수 있습니다.

```
1 app.use('/static', express.static(path.join(__dirname, 'html')));
```

위와 같이 `/static`을 넣어주면 `localhost:8080/static/turn.js` 이렇게 접근해야 합니다. 실제 경로와 브라우저의 경로를 달리 해서 쉽게 서버 파일에 접근하지 못하도록 하는 겁니다.

중간 `app.get('/', 콜백)`, `app.get('/about', 콜백)` 이 REST API로써 **라우팅**을 처리하는 부분입니다. '/'이라는 get 요청이 들어왔을 때와 '/about'이라는 get 요청이 들어왔을 때 어떻게 응답할 것인지 콜백 함수로 정의할 수 있습니다.

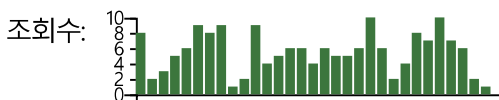
앞으로는 콜백의 매개변수로 `request`와 `response` 대신 `req`와 `res`를 쓸겁니다. `response`와 `request`는 단어가 너무 길거든요. 마지막엔 지금까지 썼던 `fs.readFile`과 `res.end` 대신 `res.sendFile` 메소드로 각각 해당하는 `html` 파일을 주면 됩니다. `html` 파일을 읽어서 브라우저로 보내는 복잡했던 코드가 한 줄로 줄어들었습니다.



다음 시간에는 Express **미들웨어**와 **고급 라우팅**에 대해 알아보겠습니다!

이전글: [HTML 전송하기](#)

다음글: [익스프레스 미들웨어, 라우팅](#)



투표로 게시글에 관해 피드백을 해주시면 게시글 수정 시 반영됩니다. 오류가 있다면 어떤 부분에 오류가 있는지도 알려주세요!
잘못된 정보가 퍼져나가지 않도록 도와주세요

목록





로그인



회원가입

오류가 있는 거 같아요

0(0.00%)

총 투표 수

30

Copyright 2016- ZeroCho. 무단 전재 및 재배포 금지. 출처 표기 시 인용 가능.

이메일

비밀번호(수정 및 삭제 시 사용)

댓글을 입력하세요. 근거없는 비난 대신 날카로운 피드백을 원합니다. 답글이 달리면 삭제할 수 없습니다.

등록

4개의 댓글이 있습니다.

- + 익명** 멍청한 질문일수도 있는데요. `const app = express();` 를 하게되면 `app`이라는 객체가 생성된다고 하셨는데 `console.log(typeof app)`을 해보면 `function`이 뜨게 됩니다. 왜 그런지 알 수 있을까요?

+ ZeroCho 함수도 객체입니다~

+ 익명 그러면 `const app = express()` 를 하게 되면 `express`라는 함수를 호출한거고 리턴값으로 함수를 받은건가요?

익명 `const app = express()`가 `listen`이라는 메서드를 사용한다면 내부에 대충 어떤식으로 구현되어있는지 밑에처럼 예를 들어주실 수 있을까요? 머리가 복잡해지네요 πππ

```
const app2 = (function(){
  return function(){}; })
```

↩ 2년 전
- + 오토미** 크 되니까 재밌네요! 별 건 아니지만 밑에서 네 번째 줄에 `response >` response

ZeroCho 감사합니다

Chon Hee Cho 좋은 설명 감사드립니다!

+ TigerStone localhost8080에서 글자가 깨져서 나오는 이유는 왜때문일까요? "영웅 이름을 입력하세요!"와 "시작" 등의 글자가 깨져서 나오네요. 여태까지 다른 실습은 결과가 잘 나왔고, 이번 Express 예제에서 "Hello

↩ 3년 전

↩ 3년 전

↩ 4년 전

↩ 5년 전

목록




페이지 좋아요

메시지 보내기

타임라인

메시지


ZeroCho Blog
 약 한 달 전
<https://www.zerocho.com/cat.../post/61d064b6a7f9490004dd9358>


ZEROCHO.COM
(etc) 2021년 ...
[2020년 블로그 ...

2개

댓글 달기

1개


ZeroCho Blog
 약 7개월 전
