package.json

 "scripts": {

  "start":"node index.js"

*/\*터미널에 작성한 명령어를 여기에 작성할 수 있음 달라진 점은 계속해서 node index.js라고 하지 않아도 되고 npm start라고 명령어를 주어도 괜찮음.*

*\*/*

“ "dependencies": // 프로그램이 실행되기위해 필요한 것.

2.4-

브라우져와 웹사이트는 Post Request로 정보를 전달한다. 영상에 코멘트를 달면 post 웅앵이 실행하게 됨. 즉 우리가 적합한 http method를 사용함.

Get request 🡺URL에 접근

URL을 쓴다(youtube.com)🡺브라우저가 Get method를 실행한다 🡺 브라우저가 URL의 페이지를 읽어온다.

웹사이트에 Login 한다. 🡺브라우저가 POST method를 실행한다 🡺 브라우저가 웹사이트에 정보를 전달한다.

서버가 응답하게 다른 사람이 접속할 수 있도록 하는 것.

const express = require('express'); *// node module을 어딘가에 가지고 옴  express를 내 파일들 속에서 찾으려고 함.  아님 없으면 node-modul에서 찾으려고 한다.*

const app = express();

const PORT=4000;

function handleListening(){

    console.log(`Listening on: http://localhost:${PORT}`); *// '와 `는 다르다!*

}

function handleHome(req,res){*//request obj response obj 누가 홈페이지를 접속하고 어떤 종류의 데이터가  페이지로 전송됐는지.. 그럴 때 req를 사용.. 아이디와 패스워드를 전송한다면 req에서 정보를 얻을 수 있다.*

    console.log(req);

    res.send("hello baby");

}

function handleProfile(req,res){

    res.send("You are on my profile");

}

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening); *// 4000개의 포트를 들으라!! app아 !!*

2.5

Babel ? 최신의 JS코드를 예전 JS코드로 전환해준다.

const something =(a,b)=>a\*b --🡪 var something = function something(a,b){return a\*b}

npm install @babel/node로 설치 해준다.

또한 많은 stage를 가지고 있음.

stage 0는 완전 실험적인 걸 얻게 될 것임.

우리가 사용할 것은 -env

<https://babeljs.io/docs/en/babel-preset-env>

babel을 적용하고 싶으면 package.json 파일에서 “start” : “babel-node index.js” 라고 바꿔준다.

그럼 babel 코드로 바뀌게 된다.

nodeman을 설치한 이유- 새로고침하고 저장하고 멈췄다가 다시 켜고.. 등등의 처리과정을 거치지 않아도 되기 때문.

2.6

middleware 란? 어려운 express 내용 처리가 끝날 때까지 기다리는 것.

어떻게 연결? 웹사이트 연결 -> route를 봄 아 !!! home으로 연결하는 구나!! -> handlehome을 실행 -> betweenHome이 middleware가 된다.

app.get("/",betweenHome,handleHome);

유저와 마지막 응답 사이에 무언ㄴ가 있어야 하는데 이를 middleware라고 함.

어떻게 middleware를 만든는지? 걍 중간에 함수 끼어놓아도 ㄱㅊ

그렇게 만들고 구글 크롬으로부터 온 요청을 계속 처리할 지에 대해 우리가 권한을 줘야함.

이는 어디에 사용되냐면 사용자의 로그인 현황을 알 수 있고 파일을 전송할 때 중간에서 가로챌 수 있다.

middleware는 오직 route에만 사용됨

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening); *// 4000개의 포트를 들으라!! app아 !!*

*}\*/*

app.use(betweenHome);

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening);

*import* express *from* "express"; *// const express = require('express');를 최신버전으로 바꿀 수 있다.*

*import* "core-js";

*// node module을 어딘가에 가지고 옴  express를 내 파일들 속에서 찾으려고 함.  아님 없으면 node-modul에서 찾으려고 한다.*

const app = express();

const PORT=4000;

const handleListening=()=> console.log(`Listening on: http://localhost:${PORT}`); *// '와 `는 다르다!*

*//섹시한 방향으로 function을 사용하지않고 const를 이용해서 바꿀 수 있다.*

const handleHome=(req,res)=>res.send("hello baby");

const handleProfile =(req,res)=> res.send("You are on my profile");

const betweenHome=(req,res,next)=>{

    console.log("between");

    next();

};

*/\*function handleProfile(req,res){*

*res.send("You are on my profile");*

*}\*/*

app.use(betweenHome);

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening); *// 4000개의 포트를 들으라!! app아 !!*

2.7

추가적인 middleware

morgan – logging 정보를 줌 (어떤 일이 일어났는지 기록하는 것)

app.use(morgan("tiny"));

profile에 들어가면 로깅한 기록이 뜨게된다.

helmet- node.js 앱의 ㅂ보안을 안전하게 만들어 줌.

이 밖에도 많은 것들이 있다…

그런데 우리가 middle ware를 사용할 때 연결을 끊을 수 있다.

cookie parser- cookie에 유저 정보를 저장.

body parser- 누가 form을 끝까지 다 채우면 form은 서버에 의해서 받아져야만 함.

이름과 비밀번호를 작성하면 받아야한다. 우리는 req ob에 접근할 수 있기를 원한다. 그러기 위해서 body parser을 이용.

const handleHome=(req,res)=>res.send("hello baby");

const handleProfile =(req,res)=> res.send("You are on my profile");

app.use(cookieparser());

app.use(bodyparser.json());

app.use(bodyparser.urlencoded({extended:true}));

app.use(morgan("div"));

app.use(helmet());

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening);

*// 4000개의 포트를 들으라!! app아 !!*

2.8

*export* *default* app;

다른 파일에서도 사용 할 수 있도록 함.

router은 route들의 복잡함을 쪼개주는데 사용할 수 있다.

userRouter.get("/",(req,res) => res.send('user index'))

userRouter.get("/edit",(req,res) => res.send('user edit'))

userRouter.get("/password",(req,res) => res.send('user password'))

이렇게 쓰기 가능

/user/edit/. …..

이처럼 router는 작은 파일로 나눠서 사용 할 수있다.

app.js

*import* { userRouter } *from* "./router";

app.use("/user",userRouter); *//누가 user경로로 접속하면 userRouter을 전체 다 사용하겠다는 의미이다.*

*export* *default* app;

router.js

*import* express *from* "express";

*export* const userRouter = express.Router(); *//이 녀석을 내보낼 수 있다.ㄴ*

userRouter.get("/",(req,res) => res.send('user index'));

userRouter.get("/edit",(req,res) => res.send('user edit'));

userRouter.get("/password",(req,res) => res.send('user password'));