

# GNB

## 웹 프로그래밍

두번째 수업 자료  
Made by 박은빈  
Copy from 양한솔

## INDEX

1. CRUD 란?
2. End to End
3. 실습

# 1. CRUD란?

- 앞으로 구현해야 할 비즈니스 로직
- 비즈니스 로직: 컴퓨터 프로그램에서 실세계의 규칙에 따라 데이터를 생성, 표시, 저장, 변경하는 부분

이름	조작
C (Create)	생성
R (Read)	조회
U (Update)	수정
D (Delete)	삭제

# 1. CRUD란?

- 이해를 위한 예시

- 회원 가입: 유저 **생성**

- 프로필 페이지: 유저 **조회**

- 닉네임 수정 혹은 비밀번호 변경: 유저 **수정**

- 회원 탈퇴: 유저 **삭제**

## 2. End to End

- 이해를 위한 사전지식 : Front end 와 Back end



- DB (DataBase)  
여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음
- API (Application Programming Interface)  
응용 프로그램에서 사용할 수 있도록, 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스  
프로그램과 또 다른 프로그램을 연결해주는 일종의 다리

## 2. End to End

- 이해를 위한 사전지식 : 라우터



- 서로 다른 네트워크 간에 중계 역할을 해준다.
- 패킷의 위치를 추출하여, 그 위치에 대한 최적의 경로를 지정하며,  
이 경로를 따라 데이터 패킷을 다음 장치로 전향 시키는 장치
- 패킷: 수행하는 데이터의 서식 단위
- 프론트에서 보낸 요청과 일치하는 함수를 찾아 줌

## 2. End to End

서버의 라우터가 생성 조직과  
매핑 된 함수를 찾는다

Front End

게시글을 입력하고  
생성요청을 보낸다

생성

조회

수정

삭제

Server.Router

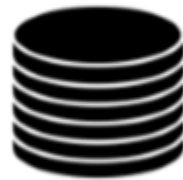
DB.create

DB.get

DB.modify

DB.remove

해당함수를 호출하여  
데이터 베이스에  
실제로 저장한다



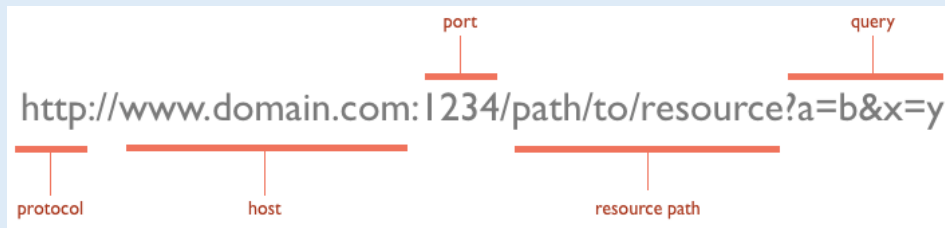
DB

Back End

### 3. 실습

- review

요청 : URI + Method



Method	Action
POST	Create (생성)
GET	Read (요청)
PUT	Update (변경)
DELETE	Delete (삭제)

### 3. 실습

- 목표1 : 메인 페이지 요청 -> Read -> GET # 오늘은 여기까지
- 목표2 : 게시물 작성 -> create -> POST
- 목표3 : 게시물 조회 -> Read -> GET
- 목표4 : 게시물 수정 -> Update -> PUT
- 목표5 : 게시물 삭제 -> Delete -> Delete



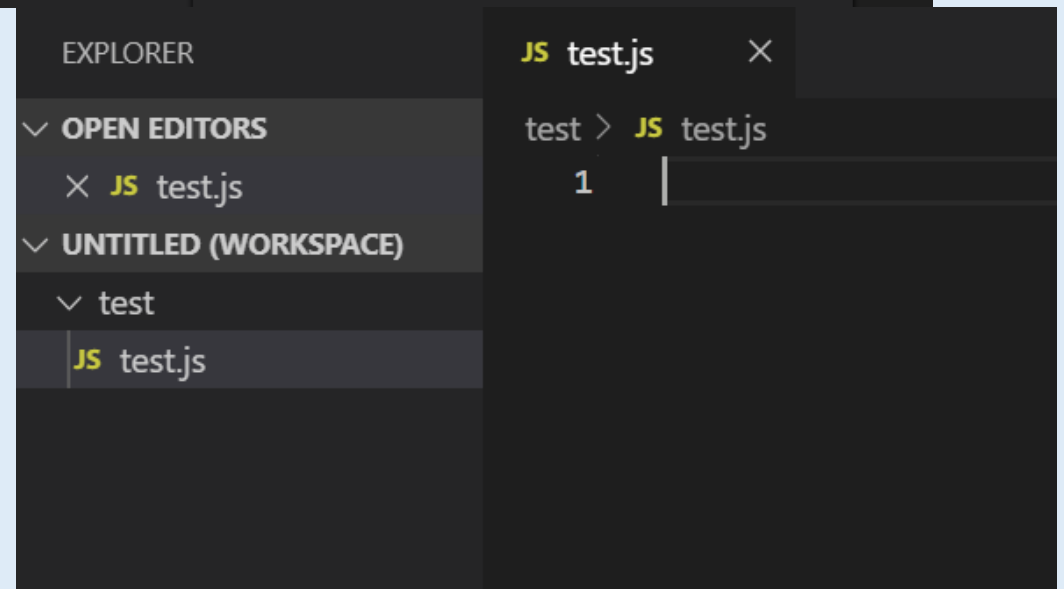
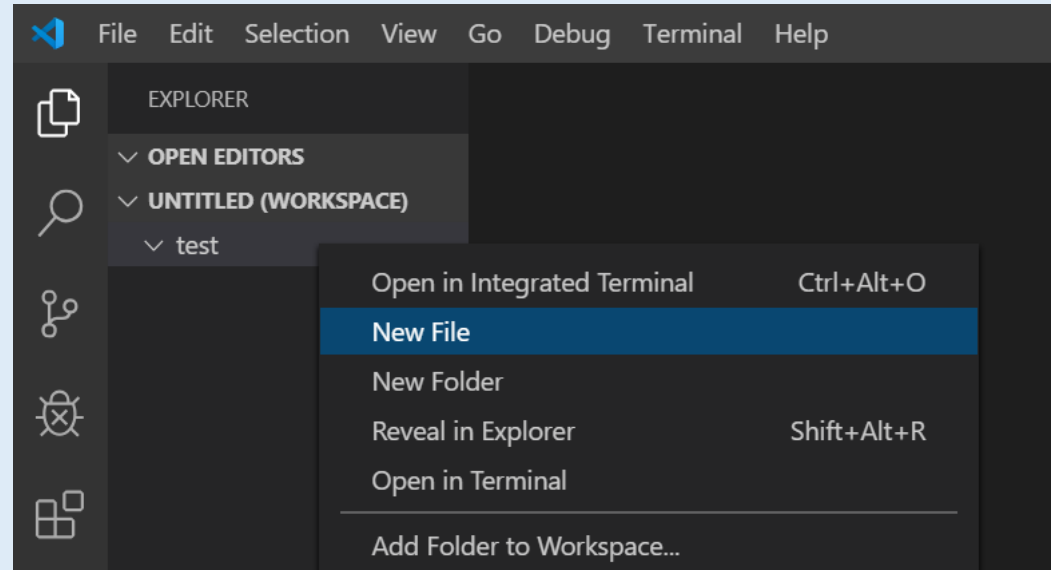
### 3. 실습

- 이해를 위한 사전 실습 : javascript

1. 아무데나 폴더 하나 만들기 (경로 알아야 함)

2. VS Code에서 폴더 열기

3. 새 파일 만들기 (파일명 : test.js)



### 3. 실습

- 이해를 위한 사전 실습 : javascript

```
test > JS test.js > ...
1 //변수생성
2 var a=10;
3
4 //콘솔에 출력
5 console.log(a);
6 //디버깅: ctrl+ F5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\Program Files\nodejs\node.exe test.js

10

```
test > JS test.js > ...
1 //함수
2 function a (){
3     var a=10;
4     console.log(a);
5 }
6
7 a();|
```

PROBLEMS DEBUG CONSOLE ...

C:\Program Files\nodejs\node.exe test.js

10

### 3. 실습

- 이해를 위한 사전 실습 : javascript

```
hellognb > routes > JS index.js > ...
```

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'Express' });
7  });
8
9  module.exports = router;
10
```

# express 가져오기-모듈 객체 생성

# Router 객체 생성-이름은 router

# router의 get (요청) 메소드

# ('path': 주소창에 '/'를 받았을 때 실행된다! , callback함수)

# callback함수: 나중에 실행하는 함수  
현재 실행되는 함수의 반환시점에 실행

# js에서 함수는 1급 객체

# argument로 함수를 넘길 수 있음

### 3. 실행

- 이해를 위한 사전 실행 : javascript

```
function(req, res){  
  res.render('index', { title: 'Express' });  
};
```

# get 요청을 하는 동안 function()실행

# req 객체는 클라이언트의 request에 대한 정보

# res 객체는 서버가 클라이언트에게 response할 정보

# res.render('view',[]) : view파일이 렌더링되어 html이 되고 html문서를 클라이언트에게 응답함

# res.render('index'): 'index.ejs'파일을 렌더링

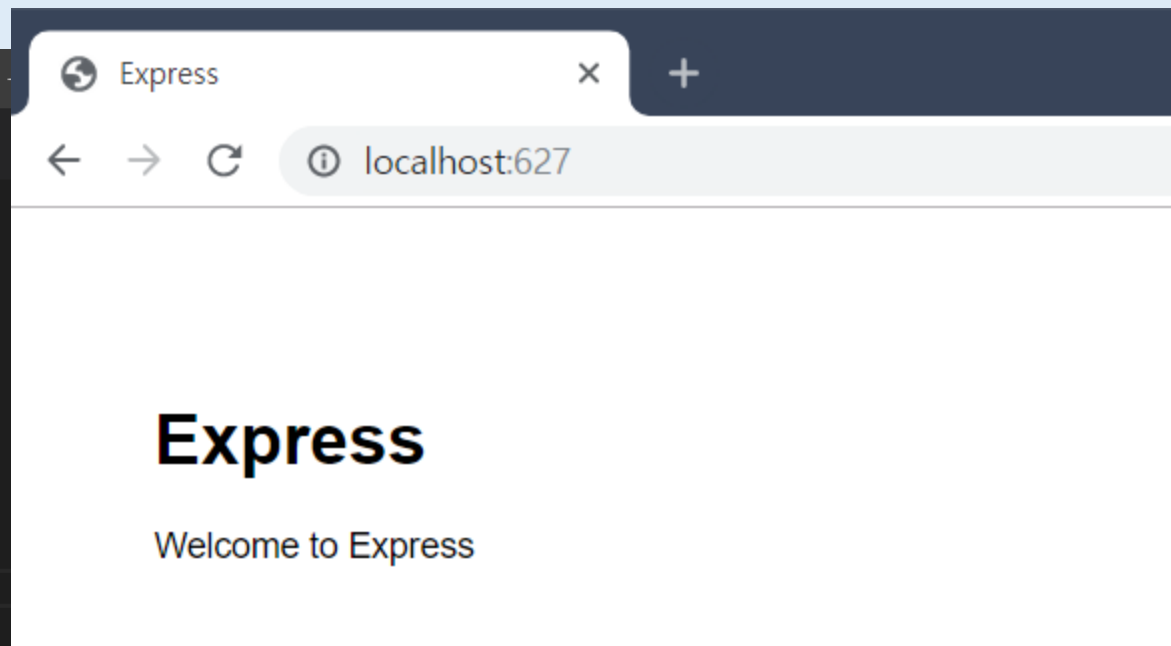
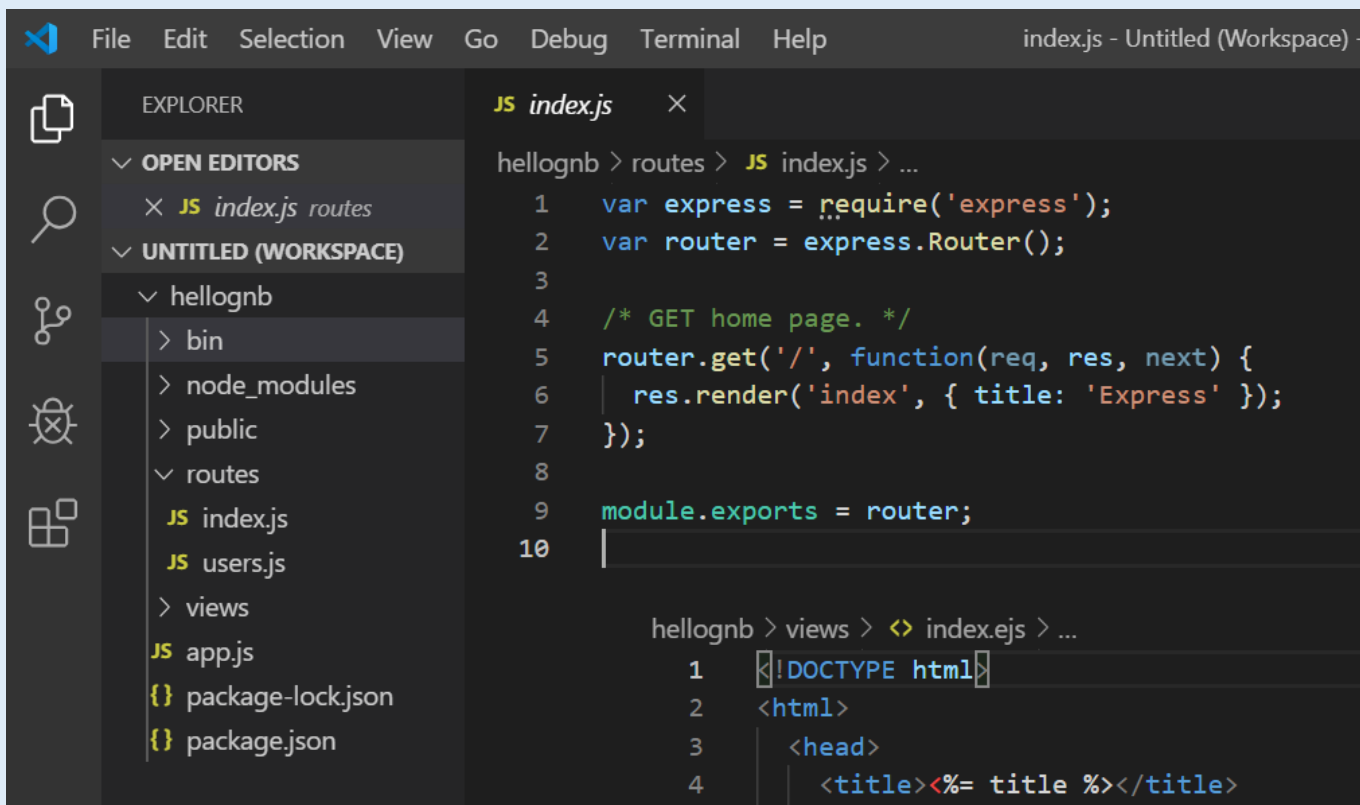
```
hellognb > views > <> index.ejs > ...  
1  <!DOCTYPE html>  
2  <html>  
3    <head>  
4      <title><%= title %></title>  
5      <link rel='stylesheet' href='/stylesheets/style.css' />  
6    </head>  
7    <body>  
8      <h1><%= title %></h1>  
9      <p>Welcome to <%= title %></p>  
10   </body>  
11  </html>  
12
```

### 3. 실습

- 응답 메소드: res.~~

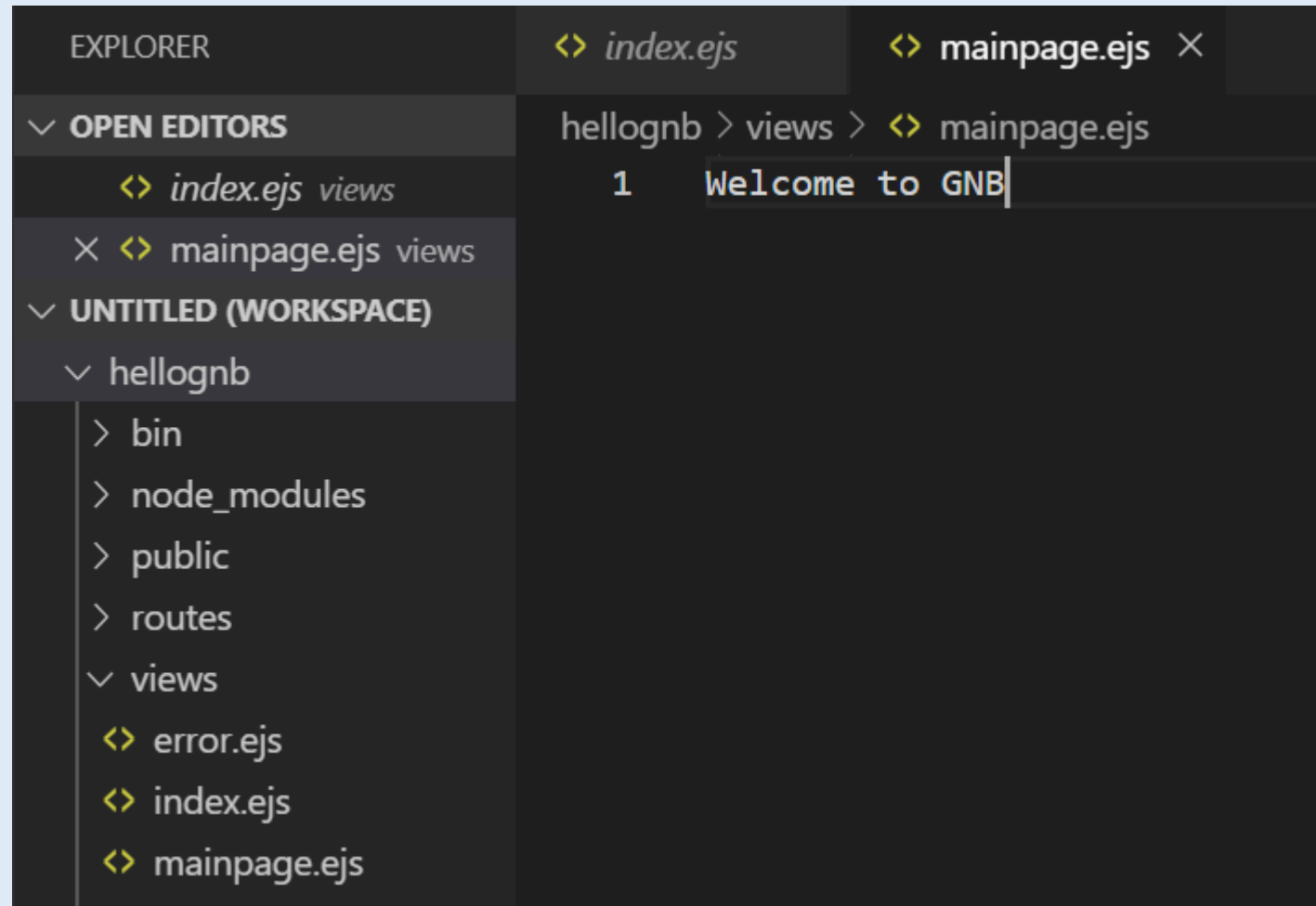
메소드	설명
<u>res.download()</u>	파일이 다운로드되도록 프롬프트합니다.
<u>res.end()</u>	응답 프로세스를 종료합니다.
<u>res.json()</u>	JSON 응답을 전송합니다.
<u>res.jsonp()</u>	JSONP 지원을 통해 JSON 응답을 전송합니다.
<u>res.redirect()</u>	요청의 경로를 재지정합니다.
<u>res.render()</u>	보기 템플릿을 렌더링합니다.
<u>res.send()</u>	다양한 유형의 응답을 전송합니다.
<u>res.sendFile</u>	파일을 옥텟 스트림의 형태로 전송합니다.
<u>res.sendStatus()</u>	응답 상태 코드를 설정한 후 해당 코드를 문자열로 표현한 내용을 응답 본문으로서 전송합니다.

### 3. 실습




### 3. 실습

- mainpage.ejs 파일을 view에 만든다
- mainpage.ejs에 Welcome to GNB를 입력 후 저장



### 3. 실습

- index.js에서 편집시작
- 원래 있던 router.get~~~지우기



```
<> mainpage.ejs  JS index.js  X
hellognb > routes > JS index.js > ...
1  var express = require('express');
2  var router = express.Router();
3
4
5
6  module.exports = router;
7
```



### 3. 실습

#### -Mission

주소창에 '/'(localhost:포트번호)가 요청 되었을 때 mainpage.ejs를 렌더링 하여 웹페이지에 띄어준다.

실행하고 크롬에서 확인

#### -hint

callback함수와 응답 메소드 render를 사용할 것.

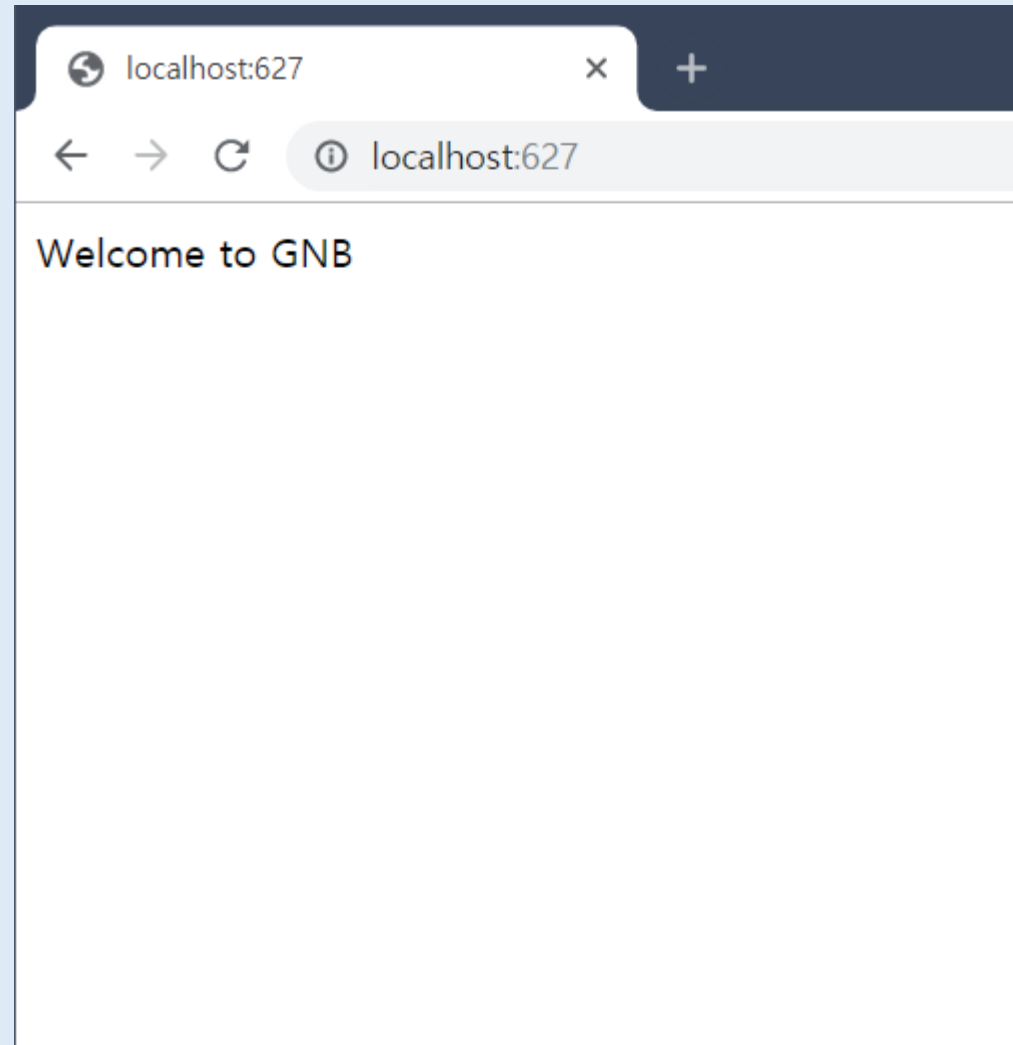
### 3. 실습

```
mainpage.ejs JS index.js ×
hellognb > routes > JS index.js > router.get('/') callback
1  var express = require('express');
2  var router = express.Router();
3
4  router.get('/',function(req,res){
5    res.render('mainpage');
6  })
7
8  module.exports = router;
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\beb99\hellognb> npm start

> hellognb@0.0.0 start C:\Users\beb99\hellognb
> node ./bin/www
```



### 3. 실습

#### -Mission

Html을 사용하여 메인 페이지를 만든다 !

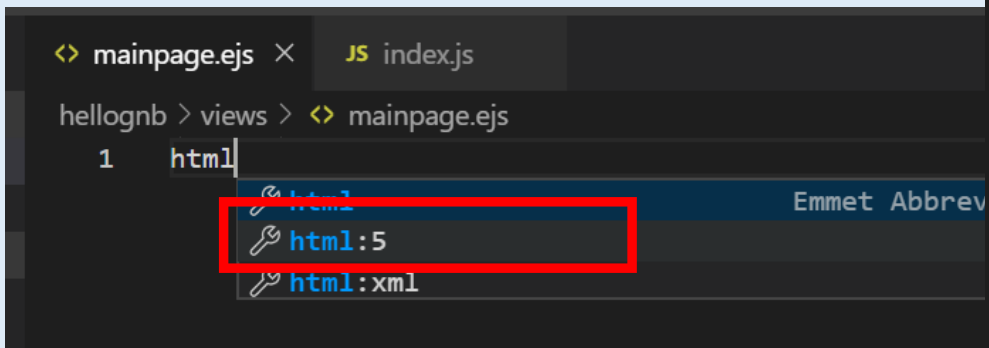
조건: title을 'GNB'로 한다.

<h1>으로 welcome to GNB

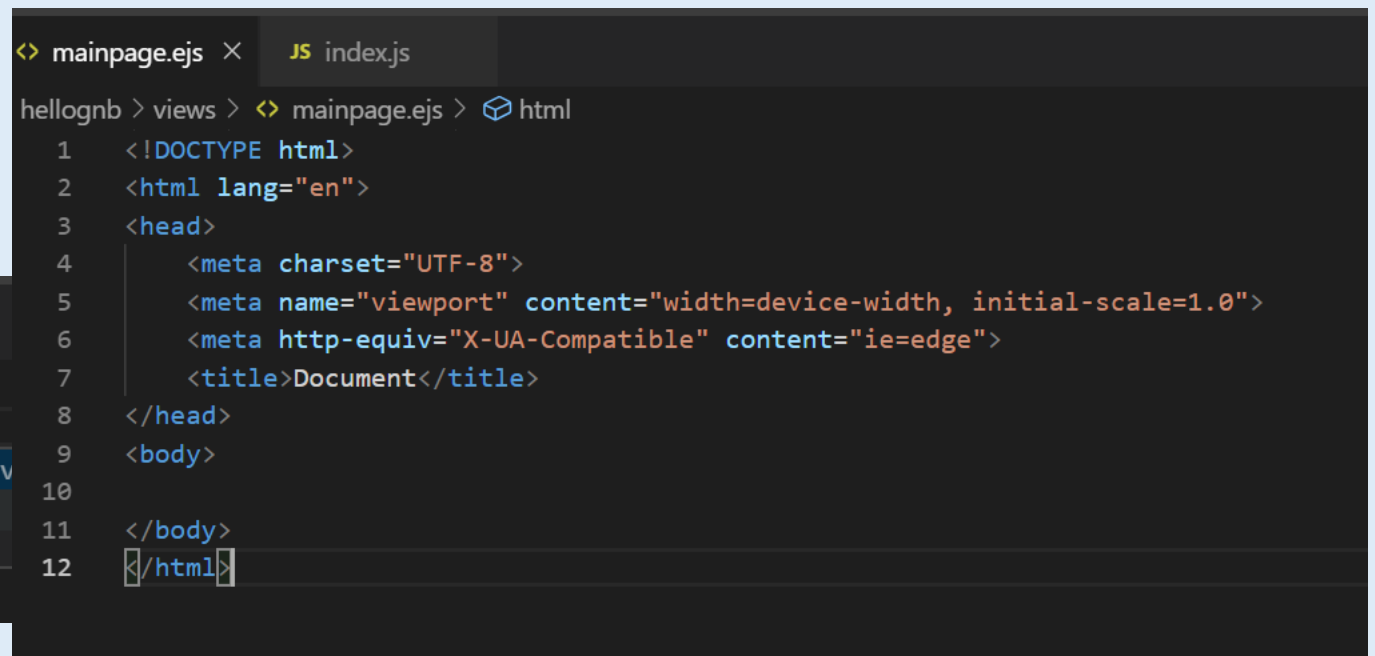
<p>로 지앤비 동아리에 오신것을 환영합니다! 지앤비는 SW동아리입니다.

#### -hint

html:5를 사용한다.




```
<> mainpage.ejs x JS index.js
hellognb > views > <> mainpage.ejs
1  html
   html
   html:5
   html:xml
```

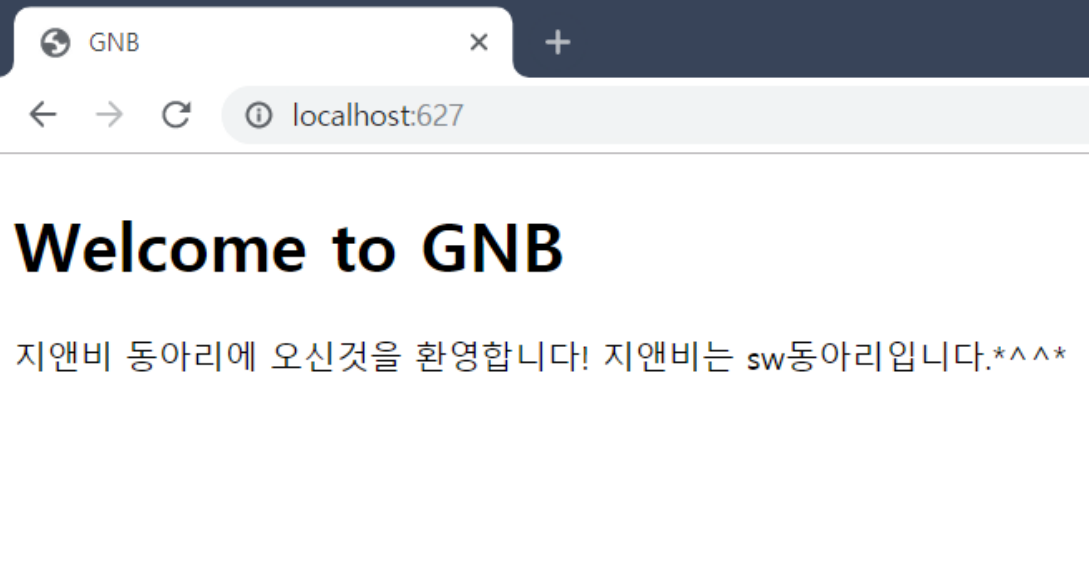


```
<> mainpage.ejs x JS index.js
hellognb > views > <> mainpage.ejs > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <meta http-equiv="X-UA-Compatible" content="ie=edge">
7    <title>Document</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

### 3. 실습

hellognb > views > <> mainpage.ejs >  html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <meta http-equiv="X-UA-Compatible" content="ie=edge">
7    <title>GNB</title>
8  </head>
9  <body>
10    <h1>Welcome to GNB</h1>
11    <p>지앤비 동아리에 오신것을 환영합니다! 지앤비는 sw동아리입니다.*^^*</p>
12  </body>
13  </html>
```



오늘은 여기까지!  
수고했어요\*^^\*