



사용자 인증을 통한 게시판 만들기

|INTRODUCE



김시훈

- 세종대학교 컴퓨터공학과
- shuni93@naver.com



황성영

- 광운대학교 컴퓨터공학과
- sy930503@naver.com

|INTRODUCE

강의자료

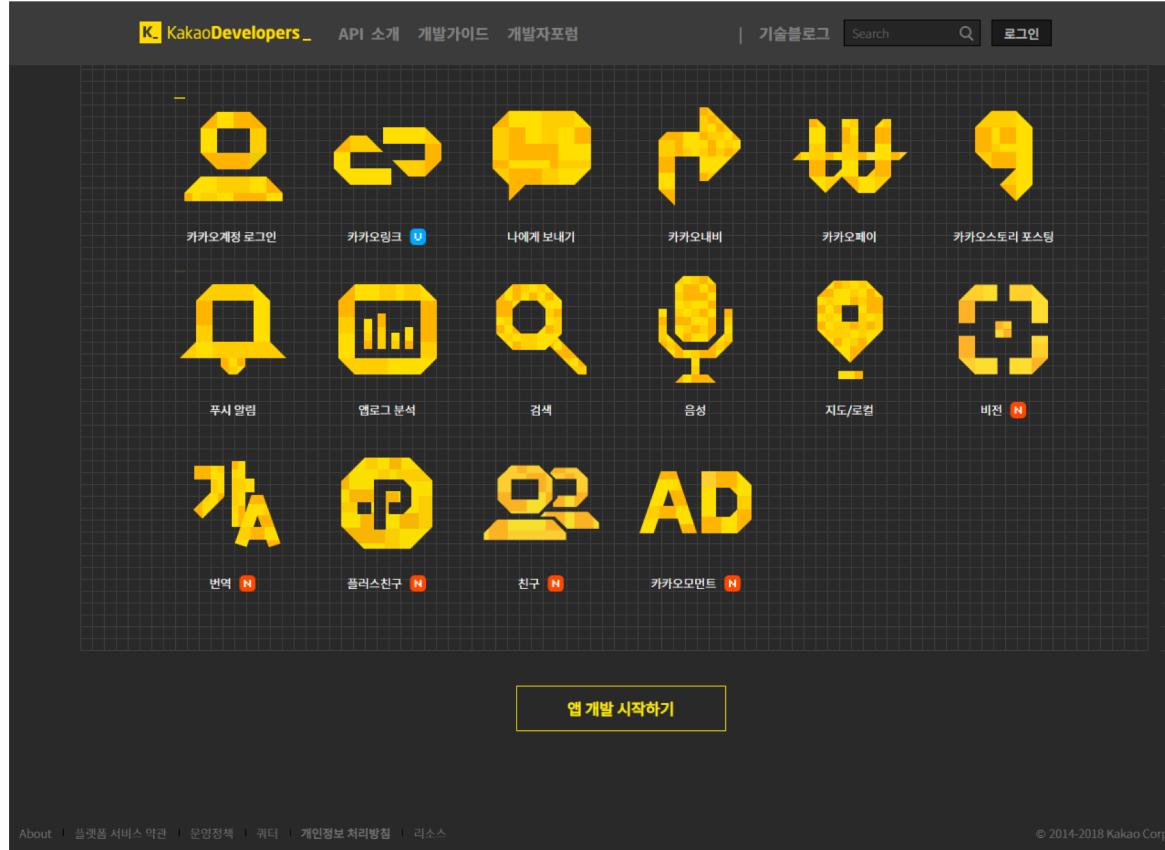
https://github.com/shuni4481/node_note

|INDEX

- ▶ #1 - Node.js의 소개 및 환경설정 / 간단한 웹 애플리케이션 만들기
- ▶ #2 - Express 모듈을 통한 프로젝트 생성 / Route / rest API
- ▶ #3 - Database(Mysql) 연동 / 회원가입 & 로그인
- ▶ #4 - 카카오톡 로그인 / CRUD 게시판 구현

|LEARN

카카오톡 로그인 인증



<https://developers.kakao.com/>

 **내애플리케이션**

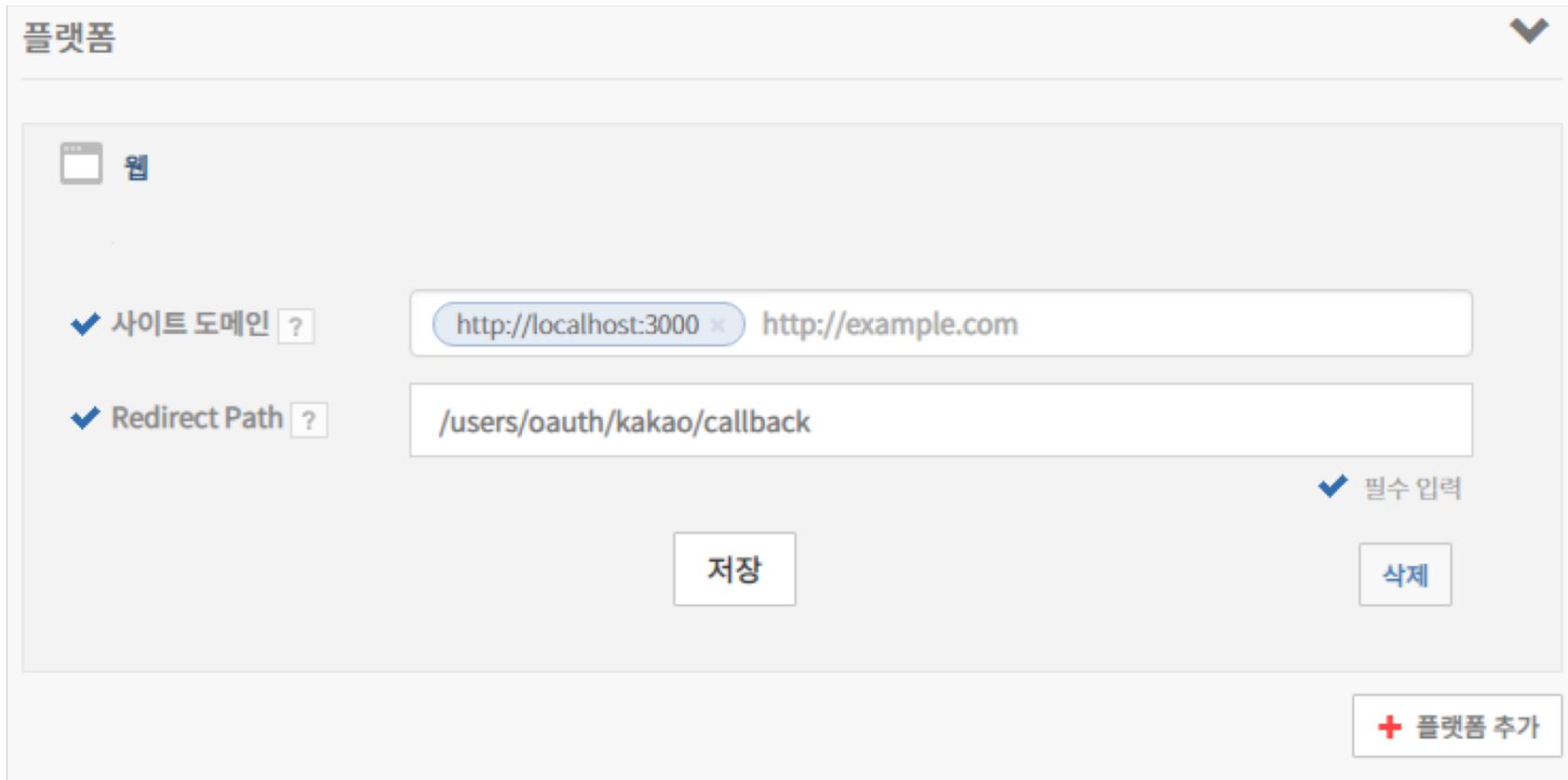
앱 만들기 ▾

앱 만들기

아이콘  **Upload**
데스크탑에서는 아이콘 이미지 파일을 끌어서 넣어도 됩니다.

이름

앱 만들기



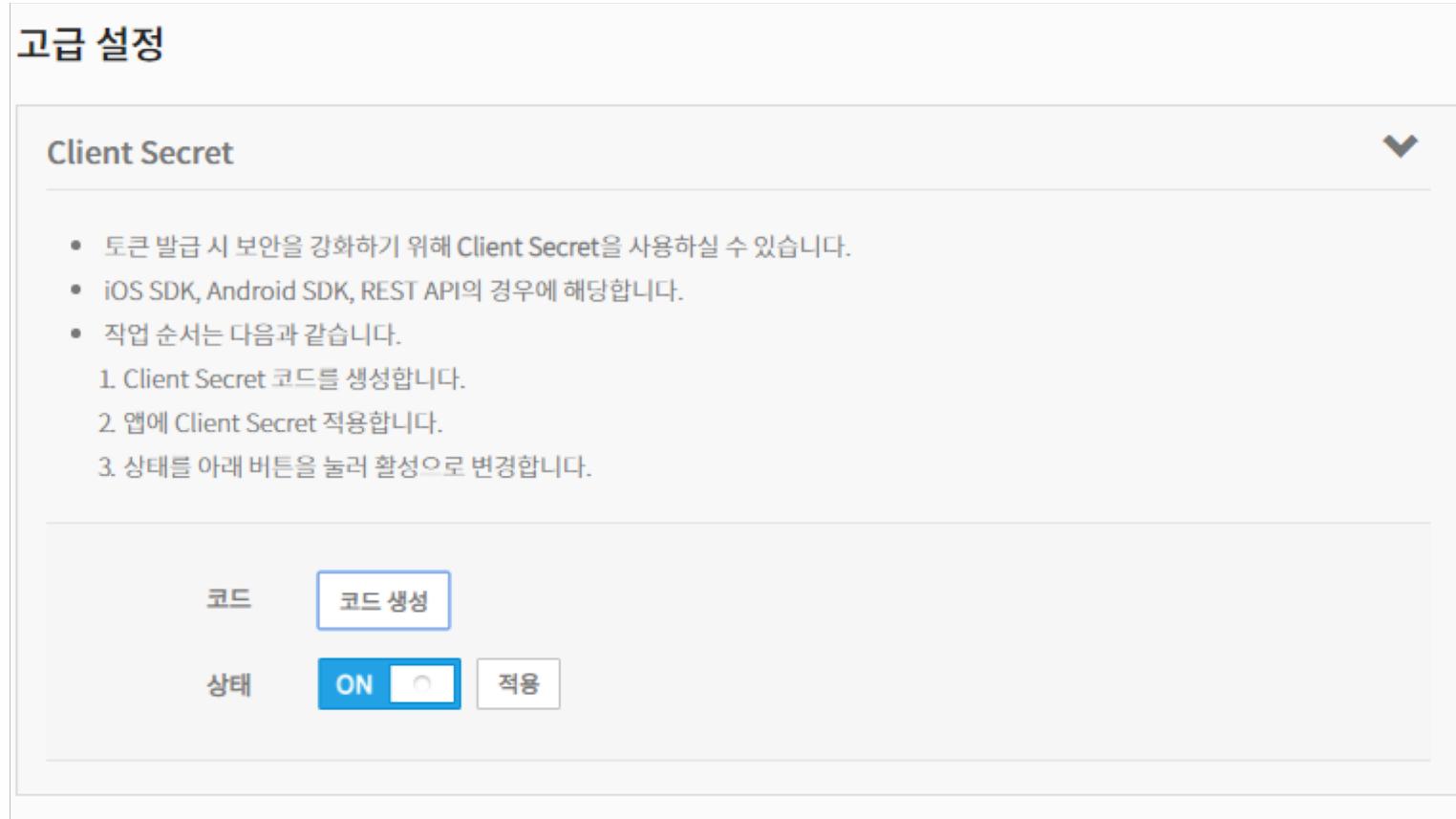
고급 설정

Client Secret

- 토큰 발급 시 보안을 강화하기 위해 Client Secret을 사용하실 수 있습니다.
- iOS SDK, Android SDK, REST API의 경우에 해당합니다.
- 작업 순서는 다음과 같습니다.
 1. Client Secret 코드를 생성합니다.
 2. 앱에 Client Secret 적용합니다.
 3. 상태를 아래 버튼을 눌러 활성으로 변경합니다.

코드 코드 생성

상태 ON 적용



사용자 관리

개발가이드: [iOS](#) [Android](#) [JavaScript](#) [REST API](#)

ON 카카오계정으로 로그인 및 API를 사용하려면 사용자 관리를 활성화하세요.

동의항목

로그인을 통해 앱(서비스) 사용자에게 보여주는 동의항목을 설정합니다.
연결 시 최소한의 동의를 받고, 특정 기능 사용 시 추가로 동의를 받을 수 있습니다.
연결 시 동의항목은 필수항목과 선택항목으로 나뉩니다.
필수항목은 사용자가 반드시 동의해야만 가입을 완료할 수 있습니다.

개인정보 보호항목 접근권한 관리항목

연결 시 필수 연결 시 선택 이용 중 사용

동의항목	Id	연결 시 필수	연결 시 선택	이용 중 사용	사용안함
프로필 정보(닉네임/프로필 사진)	profile	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

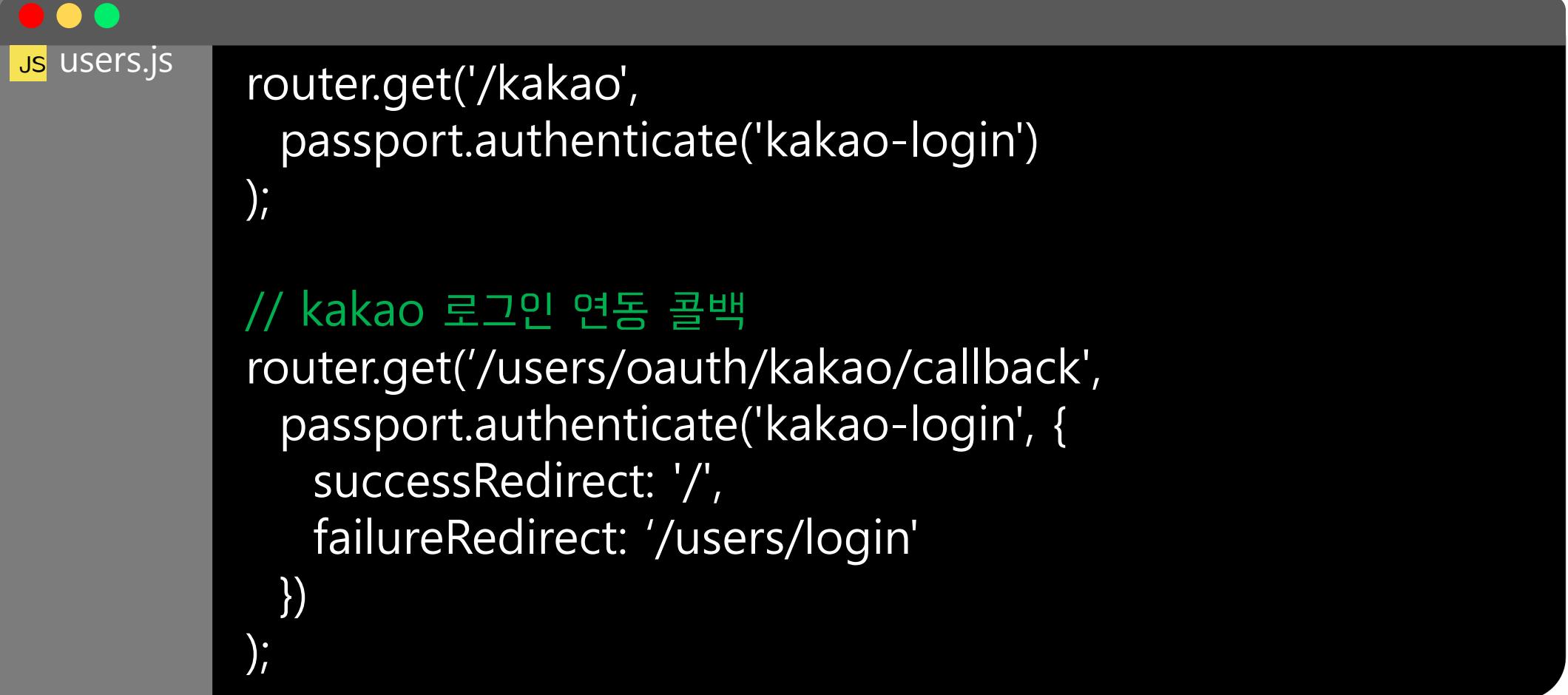
관련 API [카카오스토리 프로필 요청](#), [카카오톡 프로필 요청](#), [사용자 정보 요청](#), [앱 연결](#)

수집목적



```
npm install passport-kakao
```

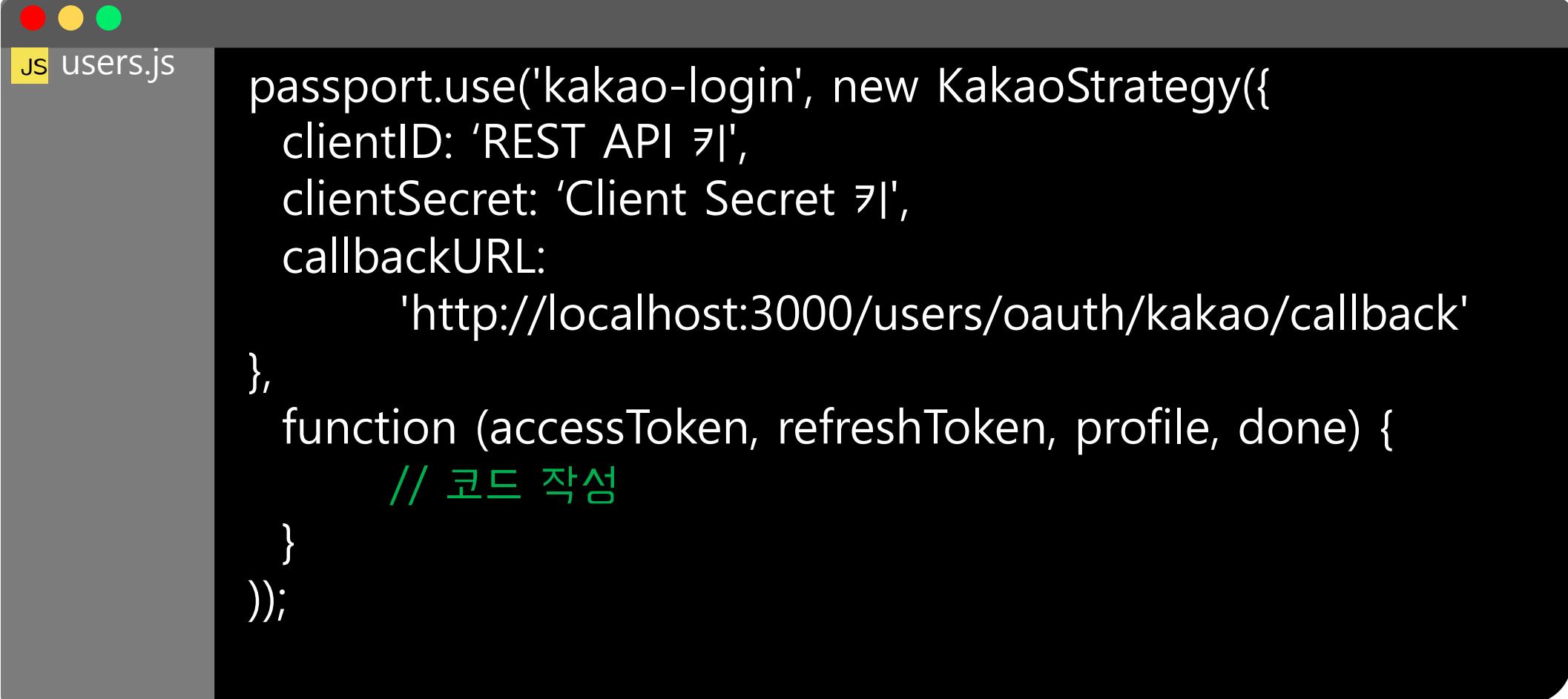
passport-kakao 모듈 설치



The screenshot shows a terminal window with a dark theme. In the top left corner, there are three small colored circles (red, yellow, green). The tab bar at the top has a yellow box containing 'JS' and the file name 'users.js'. The main area of the terminal contains the following code:

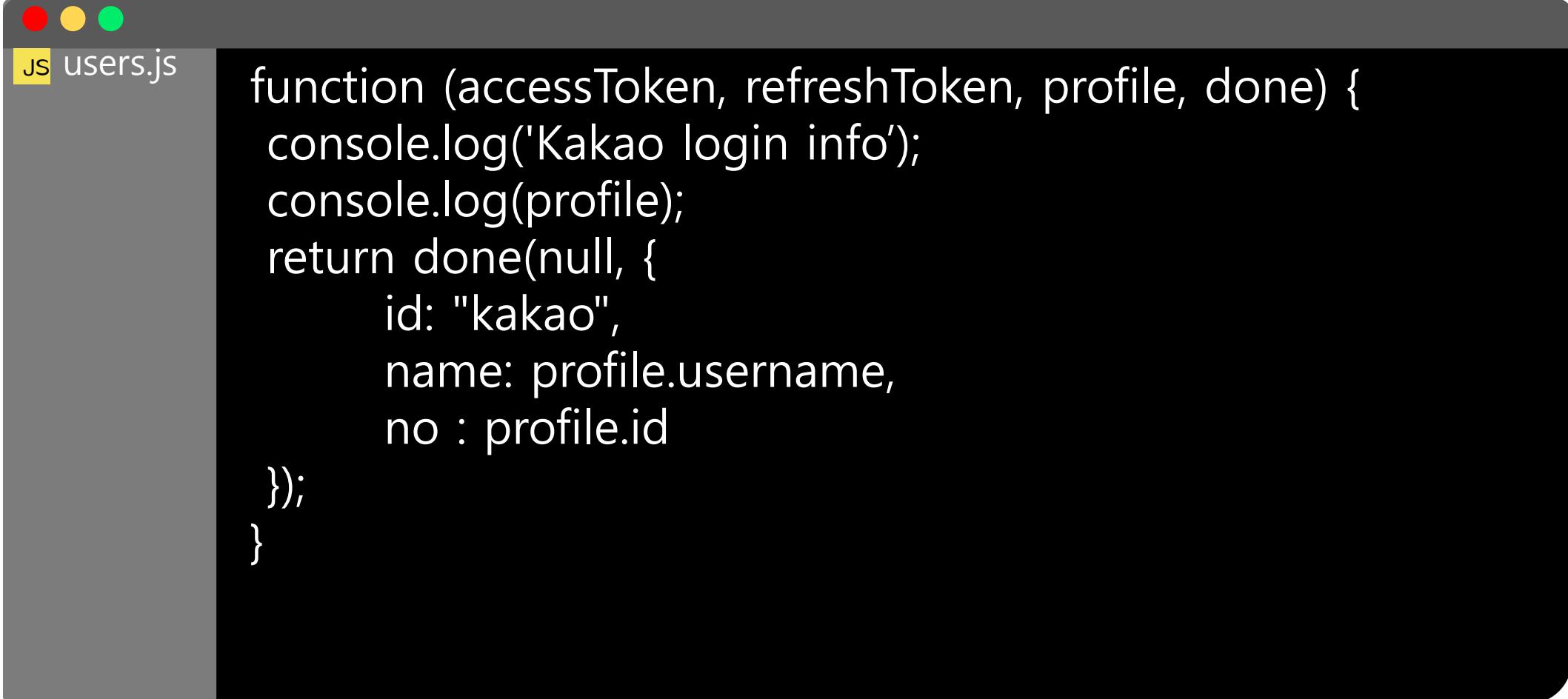
```
router.get('/kakao',
  passport.authenticate('kakao-login')
);

// kakao 로그인 연동 콜백
router.get('/users/oauth/kakao/callback',
  passport.authenticate('kakao-login', {
    successRedirect: '/',
    failureRedirect: '/users/login'
})
);
```



The screenshot shows a code editor window with a dark theme. In the top left corner, there are three circular icons: red, yellow, and green. Below them, the file name "users.js" is displayed in a yellow box. The main area contains the following JavaScript code:

```
passport.use('kakao-login', new KakaoStrategy({
  clientID: 'REST API 키',
  clientSecret: 'Client Secret 키',
  callbackURL:
    'http://localhost:3000/users/oauth/kakao/callback'
},
  function (accessToken, refreshToken, profile, done) {
    // 코드 작성
  }
));
```



The image shows a screenshot of a code editor window. The title bar indicates it's a JavaScript file named "users.js". The code itself is a function that takes four parameters: accessToken, refreshToken, profile, and done. Inside the function, two console.log statements are used to log the Kakao login information and the profile object. Then, a return statement provides the user's information in a specific format: {id: "kakao", name: profile.username, no: profile.id}. Finally, the function ends with a closing brace for the main function and another closing brace for the outermost function.

```
function (accessToken, refreshToken, profile, done) {
  console.log('Kakao login info');
  console.log(profile);
  return done(null, {
    id: "kakao",
    name: profile.username,
    no : profile.id
  });
}
```

1. 카카오 로그인으로 접속 시, 회원 가입 여부 확인
2. 신규 유저라면 회원가입 & 세션 저장 / 기존 유저이면 세션만 저장

LEARN

BoardTable 생성

b_no	b_writer	b_title	b_content	b_time
INT(11) AUTO_INCREMENT	INT(11)	VARCHAR(50)	MEDIUMTEXT	TIMESTAMP



Primary key(주 키) ->자동 증가 옵션
ex) 1, 2, 3, 4, 5 ...

```
CREATE TABLE board (
    `b_no` INT(11) NOT NULL AUTO_INCREMENT,
    `b_writer` INT(11) NOT NULL,
    `b_title` VARCHAR(50) NOT NULL,
    `b_content` MEDIUMTEXT NOT NULL,
    `b_time` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`b_no`),
    FOREIGN KEY (`b_writer`) REFERENCES `user` (`no`)
)
COLLATE='utf8_general_ci';
```

|LEARN

BoardTable 생성

```
insert into board (b_writer, b_title, b_content) values (1,  
'testTitle', 'testContent');
```

b_no	b_writer	b_title	b_content	b_time
	1	testTitle	testContent	

LEARN

Route 명세

HTTP Method	Route	View	
GET	/	index.ejs	홈화면(게시글 전체)
GET	/users	users.ejs	유저 전체 정보
GET	/users/login	login.ejs	로그인 view
POST	/users/login		로그인 처리
GET	/users/signup	signup.ejs	회원가입 view
POST	/users/signup		유저 등록
GET	/users/logout		로그아웃
GET	/users/kakao		카카오 로그인
GET	/users/oauth/kakao/callback		카카오 callback

LEARN

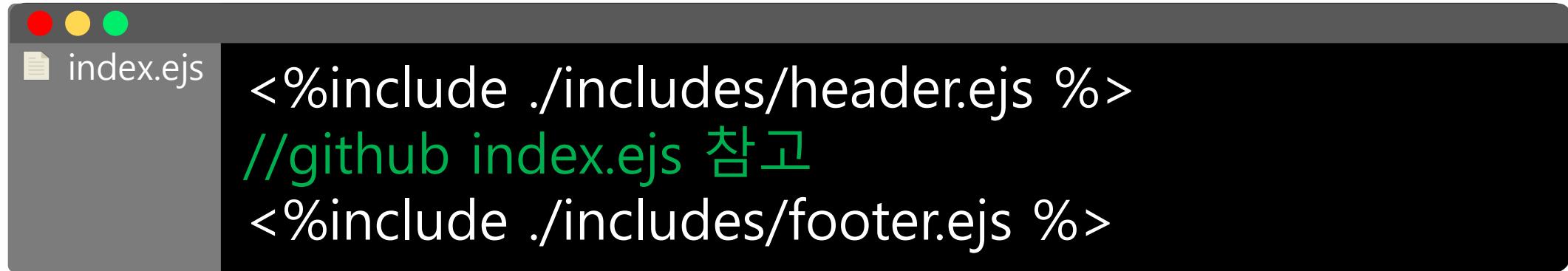
Route 명세

HTTP Method	Route	View	
GET	/notes	write.ejs	게시글 쓰기 view
POST	/notes		게시글 등록
GET	/notes/:no	detail.ejs	게시글 상세 보기
GET	/notes/update/:no	update.ejs	게시글 수정 view
POST	/notes/update/:no		게시글 수정
POST	/notes/delete/:no		게시글 삭제



index.js

```
router.get('/', function (req, res, next) {
  //글 전체 목록 조회 후 notes에 넣기
  //req.session.passport를 이용하여 로그인 안 한 사용자는
  //session:{}으로 주고 로그인 한 사용자는 req.session.passport
  res.render('index', {
    title: 'MyBoard',
    session: req.session.passport,
    notes:rows
  });
});
```



A screenshot of a code editor window titled "index.ejs". The window has a dark theme with red, yellow, and green window controls at the top left. The code editor area contains the following content:

```
<%include ./includes/header.ejs %>
//github index.ejs 참고
<%include ./includes/footer.ejs %>
```

MY BOARD

[HOME](#) [ABOUT](#)[LOGIN](#)

#	제목	작성자	작성일
5	test2	c	2019-02-11
1	123	c	2019-02-10

[글쓰기](#)

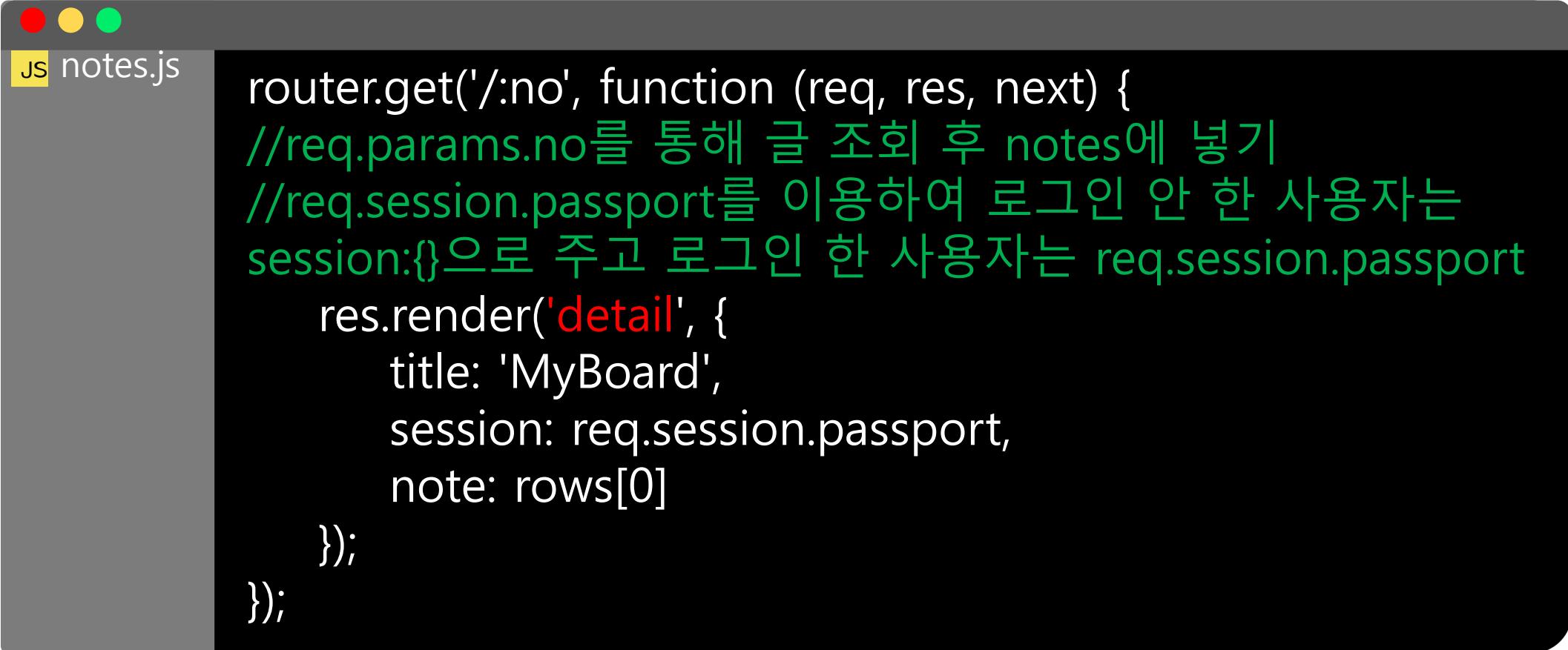
| LE



index.ejs

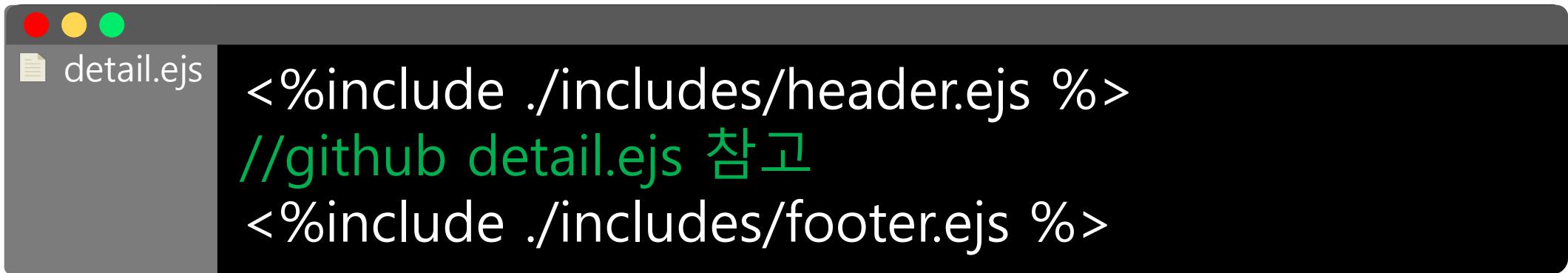
```
<script>
  document.addEventListener("DOMContentLoaded",
    function () {
      // Handler when the DOM is fully loaded
      var notes = document.getElementsByClassName('note-
lists');
      var noteHandling = function() {
        var id = this.dataset.id;
        location.href= "/notes/" + id;
      };
      for(var i =0; i<notes.length; i++) {
        notes[i].addEventListener('click', noteHandling, false);
      }
    });
  </script>
```

view



The screenshot shows a code editor window with a dark theme. At the top left, there are three circular icons: red, yellow, and green. Below them, the file name "notes.js" is displayed in a yellow box. The main area contains the following JavaScript code:

```
router.get('/:no', function (req, res, next) {
  //req.params.no를 통해 글 조회 후 notes에 넣기
  //req.session.passport를 이용하여 로그인 안 한 사용자는
  //session:{}으로 주고 로그인 한 사용자는 req.session.passport
  res.render('detail', {
    title: 'MyBoard',
    session: req.session.passport,
    note: rows[0]
  });
});
```



A screenshot of a code editor window titled "detail.ejs". The window has a dark theme with red, yellow, and green window controls at the top left. The code inside the editor is as follows:

```
<%include ./includes/header.ejs %>
//github detail.ejs 참고
<%include ./includes/footer.ejs %>
```

MY BOARD

HOME ABOUT

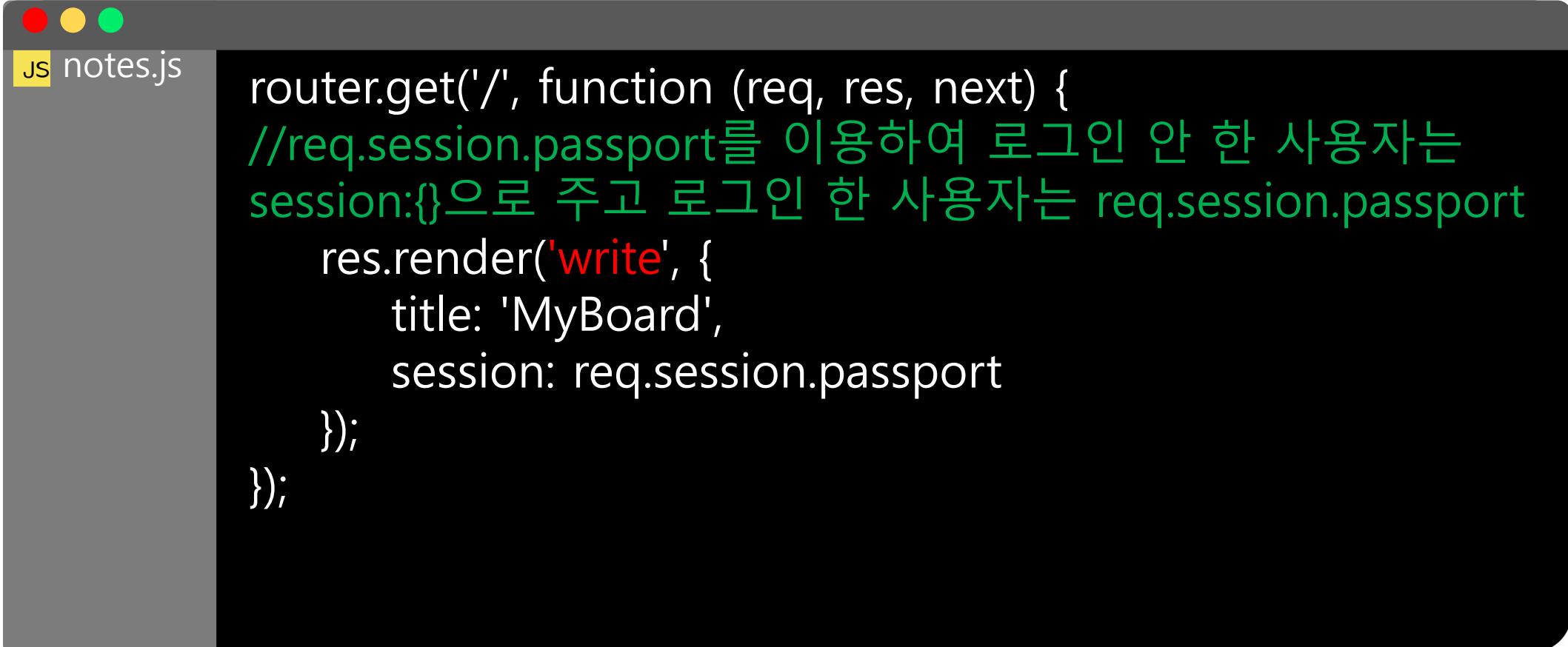
LOGIN

test2

by c | 2019-02-11

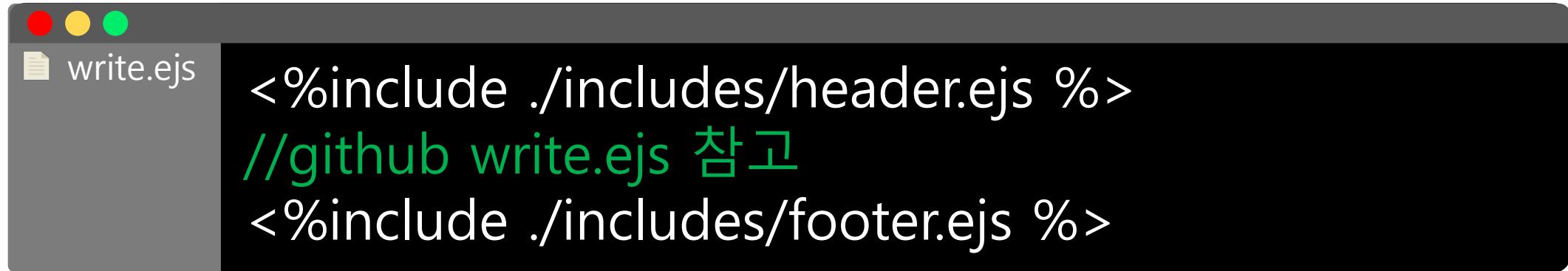
test2

목록보기 수정하기 삭제하기



The screenshot shows a code editor window with a dark theme. In the top left corner, there are three small circular icons: red, yellow, and green. Below them, the file name "notes.js" is displayed in a yellow box. The main area contains the following JavaScript code:

```
router.get('/', function (req, res, next) {
  //req.session.passport를 이용하여 로그인 안 한 사용자는
  session:{}으로 주고 로그인 한 사용자는 req.session.passport
  res.render('write', {
    title: 'MyBoard',
    session: req.session.passport
  });
});
```



A screenshot of a code editor window titled "write.ejs". The window has a dark theme with red, yellow, and green window control buttons at the top-left. The code editor area contains the following EJS template code:

```
<%include ./includes/header.ejs %>
//github write.ejs 참고
<%include ./includes/footer.ejs %>
```

Title

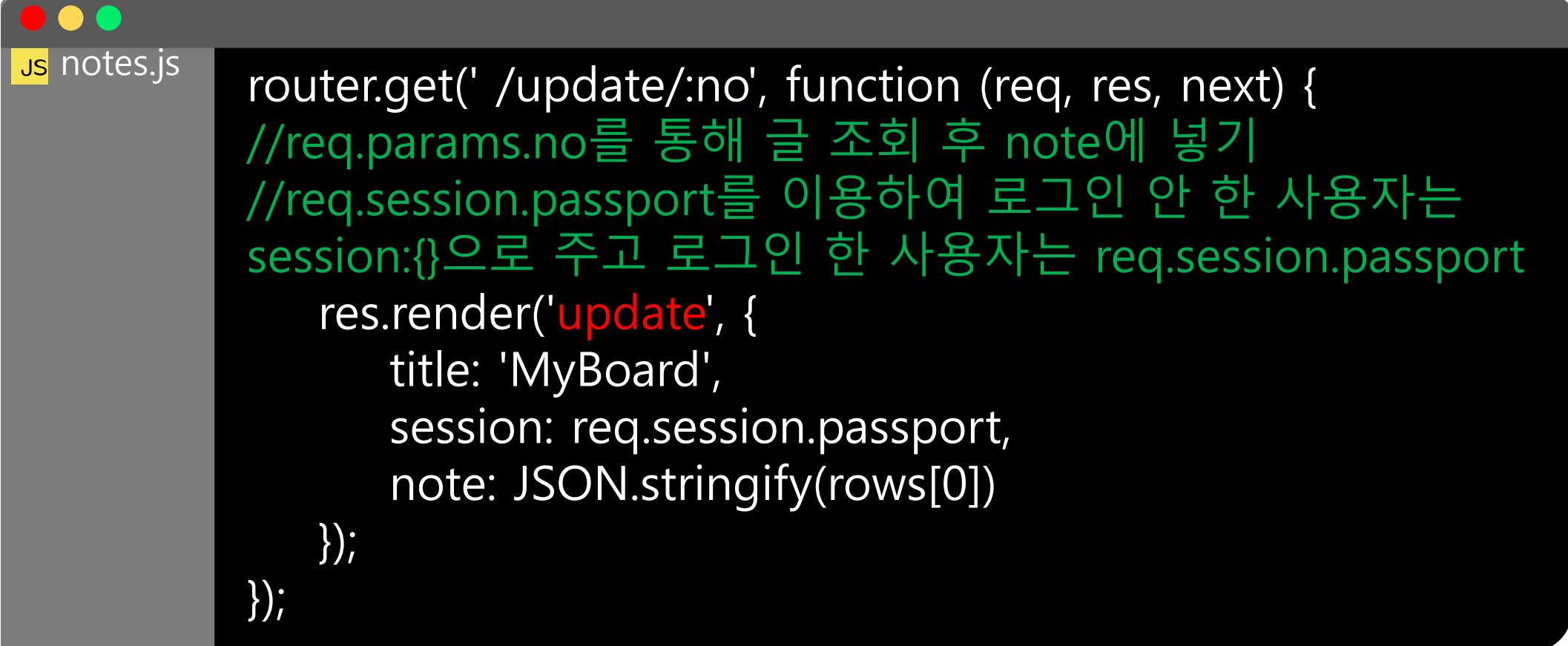
글 제목을 작성해주세요.

00자 이내로 제한됩니다.

Content

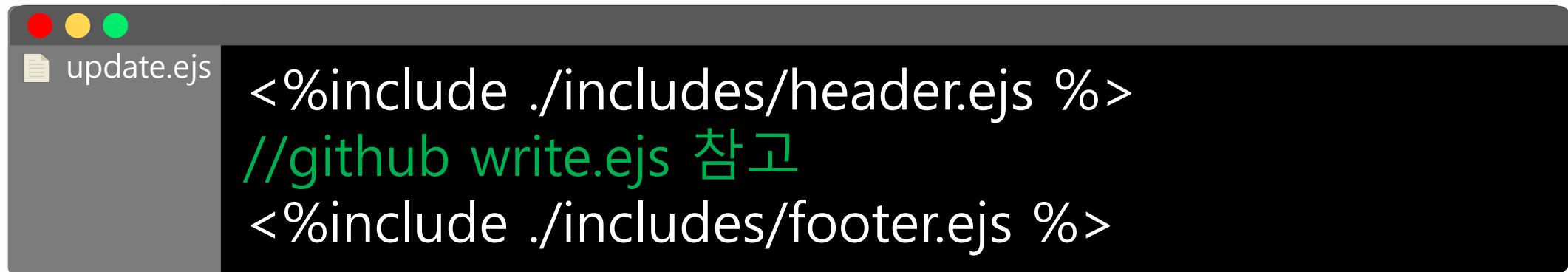
00자 이내로 제한됩니다.

작성하기



notes.js

```
router.get(' /update/:no', function (req, res, next) {
  //req.params.no를 통해 글 조회 후 note에 넣기
  //req.session.passport를 이용하여 로그인 안 한 사용자는
  //session:{}으로 주고 로그인 한 사용자는 req.session.passport
  res.render('update', {
    title: 'MyBoard',
    session: req.session.passport,
    note: JSON.stringify(rows[0])
  });
});
```



```
<%include ./includes/header.ejs %>
//github write.ejs 참고
<%include ./includes/footer.ejs %>
```



```
<script>
document.addEventListener("DOMContentLoaded",
function () {
// Handler when the DOM is fully loaded
    var note = JSON.parse('<%-note%>');
    var action = "/notes/update/" + note.b_no;
    document.getElementById('update-title').value =
note.b_title;
    document.getElementById('update-content').value =
note.b_content;
    document.getElementById("update-form").action =
action;
});
</script>
```

Title

test2

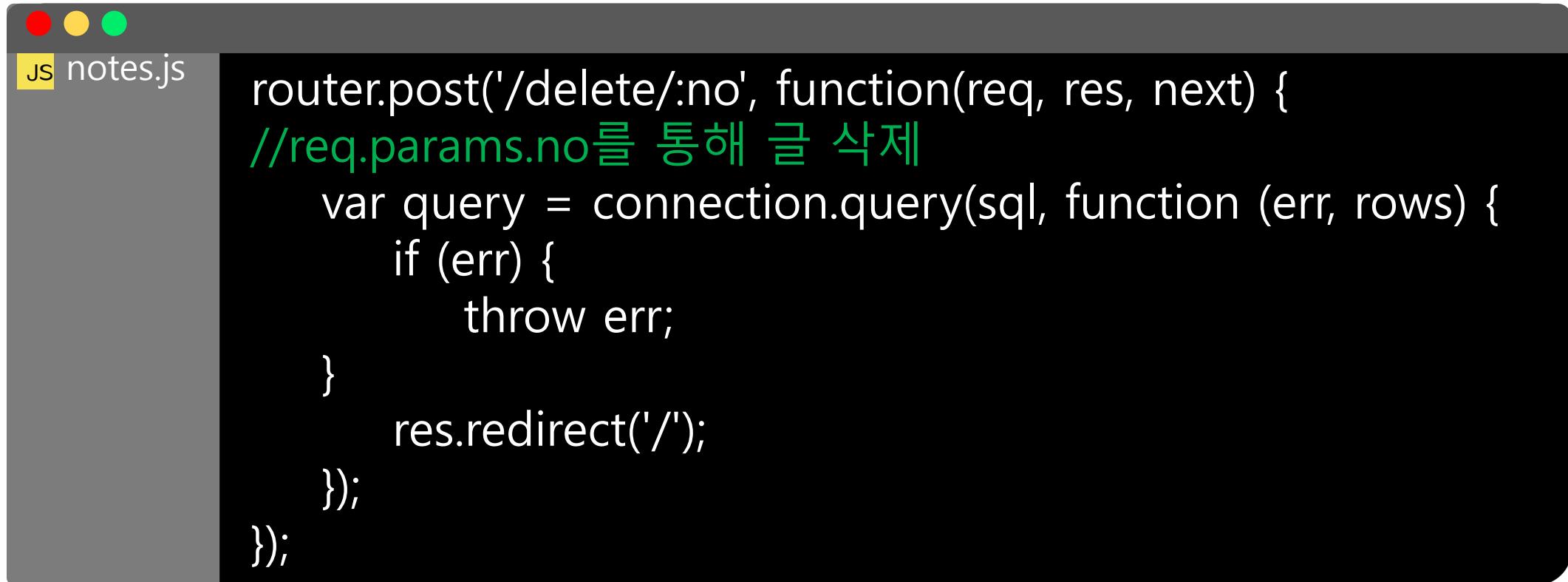
00자 이내로 제한됩니다.

Content

test2

00자 이내로 제한됩니다.

수정하기



The screenshot shows a code editor window with a dark theme. The title bar says "notes.js". The code in the editor is:

```
router.post('/delete/:no', function(req, res, next) {  
    //req.params.no를 통해 글 삭제  
    var query = connection.query(sql, function (err, rows) {  
        if (err) {  
            throw err;  
        }  
        res.redirect('/');  
    });  
});
```

The line "req.params.no를 통해 글 삭제" is highlighted in green.

THANK YOU