“이 책을 사랑해 주시는 독자 여러분 안녕하세요?

그동안 여러분이 이 책에 보내주신 소중한 의견을 모아 《Do it! Node.js 프로그래밍》을

더 편하게 공부할 수 있도록 안내하는 가이드 문서를 만들었습니다.

공부에 많은 도움이 되었으면 좋겠습니다.”

이 책으로 공부하기 전에

● 이 책은 자바스크립트를 잘 몰라도 볼 수 있습니다

이 책은 자바스크립트를 잘 몰라도 공부할 수 있도록 만들어졌습니다. 만약 자바스크립트를 잘 몰라 걱정된다면 3장을 펼쳐 보세요. 노드 제이에스를 공부하기 전에 알아두면 좋은 자바스크립트 문법을 정리해 놓았습니다.

● 이 책은 실습을 진행하며 npm을 자주 사용합니다

npm(Node Package Manager)은 노드의 모듈을 관리(설치 및 삭제)하는 프로그램입니다. 아주 간단한 구조의 서버를 만들 때는 모듈이 많이 필요하지 않겠죠? 즉, 처음부터 모든 모듈을 설치할 필요가 없습니다. 이 책은 **모듈이 필요할 때 npm으로 모듈을 설치하도록 안내합니다.** 만약 모듈을 그때그때 설치하는 것이 번거롭다면 이 문서의 **①을 참고하세요.** 각 챕터를 진행하기 전에 설치해야 하는 모듈을 정리해 놓았습니다.

▲ **모듈 설명과 함께 모듈 설치 안내** (1.png)

● 이 책의 소스코드 참고하는 방법

이 책의 본문에 실린 소스코드의 일부는 생략되어 있습니다. 만약 소스코드 전체를 확인하려면 이지스퍼블리싱 [자료실]에서 Node.js를 검색하여 실습 파일을 압축해 놓은 것(brackets_nodejs.zip)을 다운로드하세요. 그런 다음 '참조 파일'에 위치한 파일을 찾아 참고하면 됩니다.

▲ **참조 파일 위치를 보고 완성 파일 찾기** (2.png)

● 수정된 소스코드 안내

업데이트된 모듈을 사용하기 위하여 일부 소스코드가 수정되었습니다. 자세한 내용은 이 문서의 **②를 참고하세요.**

Chapter 4 → ch04_test15.js 파일 (본문 129쪽)
Chapter 7 → route_loader.js 파일 (본문 314쪽)

● 정오표를 확인하세요

여러분이 질문한 내용을 정오표에 정리해 두었습니다. 다음은 본문을 보완한 쪽수를 정리한 표입니다. 자세한 내용은 이 문서의 **③을 참고하세요.**

31 page	설명 수정	**145 page**	설명 추가
78 page	설명 수정	**149 page**	설명 수정
84 page	설명 추가	**164 page**	코드 수정
94 page	설명 추가	**168 page**	설명 추가
105 page	코드 수정	**262 page**	오탈자
107 page	설명 추가	**276 page**	설명 추가
109 page	설명 삭제	**348 page**	코드 수정
119 page	표의 코드 수정	**373 page**	코드 수정/ 설명 추가
127 page	코드 수정	**427 page**	오탈자
136 page	코드 수정	**435 page**	오탈자
137 page	코드 수정		

① 각 장별로 설치해야 하는 모듈 목록(Node.js 10.13.0 LTS 기준)

필요할 때마다 npm으로 모듈을 설치하는 것이 번거롭다면 다음을 참고하세요. 새로운 장을 시작하기 전에 명령 프롬프트를 열고 brackets-nodejs₩NodeExample1 폴더로 이동한 다음 목록에 표시된 모듈을 설치하면 됩니다.

● 참고 페이지: 37쪽 / 51쪽 / 73~74쪽

==========

Chapter 2
npm install nconf

Chapter 4
npm install winston
npm install winston-daily-rotate-file

Chapter 5
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install multer
npm install cors

Chapter 6
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongodb
npm install mongoose
npm install mysql

Chapter 7
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose

Chapter 8
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install pug

Chapter 9
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install passport
npm install passport-local
npm install passport-facebook
npm install passport-twitter
npm install passport-google-oauth
npm install connect-flash

Chapter 10
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install passport
npm install passport-local
npm install passport-facebook
npm install passport-twitter
npm install passport-google-oauth
npm install connect-flash
npm install socket.io
npm install cors

Chapter 11
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install passport
npm install passport-local
npm install passport-facebook
npm install passport-twitter
npm install passport-google-oauth
npm install connect-flash
npm install cors
npm install jayson
npm install crypto-js

Chapter 12
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install passport
npm install passport-local

```
npm install passport-facebook
npm install passport-twitter
npm install passport-google-oauth
npm install connect-flash
npm install cors
```

Chapter 13

```
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install passport
npm install passport-local
npm install passport-facebook
npm install passport-twitter
npm install passport-google-oauth
npm install connect-flash
npm install cors
npm install node-gcm
```

Chapter 14

```
npm install express
npm install express-error-handler
npm install cookie-parser
npm install body-parser
npm install express-session
npm install mongoose
npm install ejs
npm install passport
npm install passport-local
npm install passport-facebook
npm install passport-twitter
npm install passport-google-oauth
npm install connect-flash
npm install cors
npm install html-entities
```

② 수정된 소스코드

다음은 모듈이 업데이트되면서 수정된 소스코드입니다. 본문 쪽수와 함께 수정된 소스코드의 위치를 적어두었습니다.

==========

Chapter 4 → ch04_test15.js 파일 소스 코드 참조(본문 129쪽)

```
var winston = require('winston');
var winstonDaily = require('winston-daily-rotate-file');

var logger = winston.createLogger({
    level: 'debug',
    format: winston.format.simple(),
    transports: [
      new (winston.transports.Console)({
        colorize: true
      }),
      new (winstonDaily)({
        filename: './log/server_%DATE%.log',
        maxSize: '10m',
        datePattern: 'YYYY-MM-DD'
      })
    ]
});


var fs = require('fs');

var inname = './output.txt';
var outname = './output2.txt';

fs.exists(outname, function (exists) {
    if (exists) {
        fs.unlink(outname, function (err) {
            if (err) throw err;
            logger.info('기존 파일 [' + outname +'] 삭제함.');
        });
    }

    var infile = fs.createReadStream(inname, {flags: 'r'} );
    var outfile = fs.createWriteStream(outname, {flags: 'w'});

    infile.pipe(outfile);
    logger.info('파일 복사 [' + inname + '] -> [' + outname + ']');
});
```

Chapter 7 → route_loader.js 파일 소스 코드 참조(본문 314쪽)

```
var route_loader = {};

var config = require('../config');


route_loader.init = function(app, router) {
        console.log('route_loader.init 호출됨.');
        return initRoutes(app, router);
}

// route_info에 정의된 라우팅 정보 처리
function initRoutes(app, router) {

        var infoLen = config.route_info.length;
        console.log('설정에 정의된 라우팅 모듈의 수 : %d', infoLen);

        for (var i = 0; i < infoLen; i++) {
        var curItem = config.route_info[i];

        // 모듈 파일에서 모듈 불러옴
        var curModule = require(curItem.file);
        console.log('%s 파일에서 모듈정보를 읽어옴.', curItem.file);

        //  라우팅 처리
        if (curItem.type == 'get') {
                router.route(curItem.path).get(curModule[curItem.method]);
        } else if (curItem.type == 'post') {
                router.route(curItem.path).post(curModule[curItem.method]);
        } else {
        router.route(curItem.path).post(curModule[curItem.method]);
        }


        console.log('라우팅 모듈 [%s]이(가) 설정됨.', curItem.method);
        }

    // 라우터 객체 등록
    app.use('/', router);
}

module.exports = route_loader;
```

③ 정오표(2018년 11월 기준)

지금까지 확인된 오탈자 및 내용 오류를 수정한 정오표입니다. 순서대로 작성된 페이지를 찾아가면 쉽게 확인할 수 있습니다.

==========

31 page

이런 문제를 해결하기 위해 만든 게 노드입니다.

→

노드에서는 비동기 입출력을 이용해 이런 문제를 해결합니다.

78 page

파일 패스에서 파일의 확장자를 제외한 이름을 반환합니다.

→

파일 패스에서 파일 이름을 반환합니다. 만약 두 번째 파라미터로 확장자를 전달하는 경우에는 확장자를 제외한 파일 이름을 반환합니다.

84 page

문자열은 큰따옴표(", ") 또는 작은따옴표(', ')를 사용하여 표기합니다.

→

문자열은 큰따옴표(", ") 또는 작은따옴표(', ')를 사용하여 표기합니다. 일반적으로는 작은따옴표를 권장하지만 경우에 따라서는 큰따옴표를 사용하기도 합니다. 특히 문자열 내에 작은따옴표가 들어가야 하는 경우에는 큰따옴표를 사용하면서 그 안에 작은따옴표를 넣는 경우가 많습니다.

94 page

이 파일을 실행하면 다음과 같이 사용자 객체의 수가 3으로 출력됩니다.

→

이 파일을 실행하면 다음과 같이 사용자 객체의 수가 3으로 출력됩니다. push 메소드를 이용해 추가한 객체는 배열의 가장 마지막에 추가되었으며 이로 인해 배열 원소의 개수는 3이 되었습니다.

105 page [코드 내부 수정]

console.log(person01.name + '객체의 walk(10)을 호출합니다.');

→

console.log(person01.name + ' 객체의 walk(10)을 호출합니다.');

107 page

실제 인스턴스 객체를 만들 때 메모리를 효율적으로 관리할 수 있게 됩니다.

→

실제 인스턴스 객체를 만들 때 메모리를 효율적으로 관리할 수 있게 됩니다.
프로토타입에 대한 좀 더 구체적인 내용을 알고 싶다면 자바스크립트 입문서를 참고하세요.

109 page

이 작업을 쉽게 할 수 있도록 노드에 미리 만들어 둔 모듈이 **url 모듈**입니다.

→

[이 내용은 삭제 바랍니다.]

119 page [표 내의 코드 수정됨]

메소드 이름	설명
~~readFile(filename, [encoding], [callback])~~	비동기식 IO로 파일을 읽어 들입니다.
~~readFileSync(filename, [encoding])~~	동기식 IO로 파일을 읽어 들입니다.
~~writeFile(filename, data, encoding='utf8', [callback])~~	비동기식 IO로 파일을 씁니다.
~~writeFileSync(filename, data, encoding='utf8')~~	동기식 IO로 파일을 씁니다.

→

메소드 이름	설명
readFile(path[, options], callback)	비동기식 IO로 파일을 읽어 들입니다.
readFileSync(path[, options])	동기식 IO로 파일을 읽어 들입니다.
writeFile(file, data[, options], callback)	비동기식 IO로 파일을 씁니다.
writeFileSync(file, data[, options])	동기식 IO로 파일을 씁니다.

127 page [코드 내부 수정]

```
중략...
fs.exists(outname, function(exists) {
  if(exists) {
    fs.unlink(outname, function(err) {
      if(err) throw err;
      console.log('기존 파일 [' + outname +'] 삭제함.');
    });
  }
중략...
```

→

```
중략...
fs.exists(outname, function(exists) {
  if(exists) {
    fs.unlink(outname, function(err) {
      if(err) throw err;
      console.log('기존 파일 [' + outname +'] 삭제함.');
    });

    return;
  }
중략...
```

136 page

listen(port[, hostname][, backlog][, callback]

→

listen([port[, host[, backlog]]][, callback])

137 page [코드 내부 수정]

중략...
server.listen(port, host, '50000', **function**() {
중략...

→

중략...
server.listen(port, host, 50000, **function**() {
중략...

145 page

헤더를 설정할 수 없는 등의 제약이 생기므로 필요할 때만 사용하길 권합니다.

→

헤더를 설정할 수 없는 등의 제약이 생기므로 필요할 때만 사용하길 권합니다.
만약 헤더를 추가로 설정하고 싶다면 res.writeHead 메소드를 호출하여 설정할 수 있습니다.

149 page

GET 방식은 헤더 부분에 요청 정보들을 넣어 보내고

→

GET 방식은 요청 URL에 요청 정보들을 추가하여 보내고

164 page

static 미들웨어

→

serve-static 미들웨어

static 미들웨어는

→

serve-static 미들웨어는

164 page [코드 내부 수정]

```
var static = require('serve-static');
중략...
app.use('/public', static(path.join(__dirname, 'public')));
```

→

```
var serveStatic = require('serve-static');
중략...
app.use('/public', serveStatic(path.join(__dirname, 'public')));
```

```
app.use('/public', static(path.join(__dirname, 'public')));
```

→

```
app.use('/public', serveStatic(path.join(__dirname, 'public')));
```

static( ) 메소드를

→

serveStatic( ) 메소드를

static( ) 함수로

→

serveStatic( ) 함수로

168 page

login.html 문서에 접근할 때는 호출되지 않습니다.

→

login.html 문서에 접근할 때는 호출되지 않습니다.
이 문서는 serve-static 모듈을 통해 열려 있는 [public] 폴더 안에 들어있어 serve-static 미들웨어에서 응답으로 보내주게 되며 이 때문에 그 이후에 등록된 미들웨어는 호출되지 않습니다.

262 page

I accept th license terms

→

I accept the license terms

276 page

?? 또는 ? 기호를 대체한 후 다음과 같은 SQL문을 만들어 냅니다.

→

?? 또는 ? 기호를 대체한 후 다음과 같은 SQL문을 만들어 냅니다.
전달된 파라미터 값을 SQL문 안에 있는 기호와 바꿀 때 ?? 기호는 그대로 바꾸고 ? 기호는 작은따옴표를 붙이면서 바꾸게 됩니다. 따라서 SQL문의 where절 안에 들어갈 값 중에서 작은따옴표가 붙어야 하는 문자열 값에는 ? 기호를 사용하고 작은따옴표가 붙지 않는 숫자 값에는 ?? 기호를 사용합니다.

348 page [코드 내부 수정]

중략...
<h2><% = title %></h2>
중략...

→

중략...
<h2><%= title %></h2>
중략...

373 page [코드 내부 수정]

중략...
passport.use(**new** LocalStrategy(
중략...

→

중략...
passport.use('local', **new** LocalStrategy(
중략...

passport 객체의 use( ) 메소드를 사용하면 스트래티지를 설정할 수 있습니다.

→

passport 객체의 use( ) 메소드를 사용하면 스트래티지를 설정할 수 있습니다. use 메소드를 호출할 때는 두 개의 파라미터를 전달할 수 있는데 첫 번째는 참조할 이름, 두 번째는 스트래티지 객체입니다. 참조할 이름은 생략할 수 있으며 생략한 경우에는 스트래티지 객체에 설정한 name 속성 값을 참조하게 됩니다.

427 page

attach(httpServer, options)
listen(httpServer, options)

→

attach(httpServer[, options])
listen(httpServer[, options])

435 page [코드 내부 수정]

recepient

→

recipient