# 네트워크서비스 프로토콜

# 13주차 1


» 소프트웨어학부
» 김형균 교수

# 수업에 들어가며


- 지난 시간 복습
  - Express 미들웨어
  - 미들웨어의 유형
  - 기존 미들웨어 사용하기 body-parser
  - 기존 미들웨어 사용하기 compression
  - Express 미들웨어 만들기
- 오늘 학습할 내용
  - 익스프레스 에러처리
  - pageId 입력 오류 처리
  - express.Router 클래스
  - express.Router사용-라우터 정리
  - 라우터를 만들어서 파일로 분리
  - - topic.js 생성
  - - index.js 생성
  - - main.js 수정
  - Express generator

# 미들웨어의 유형



## Application

- 어플리케이션 전역에서 처리가 가능한 미들웨어로 어플리케이션 자체에 request가 발생할 때마다 실행됩니다.

## Router

- 동작방식은 Application-level 미들웨어랑 같습니다. 하지만 라우터 단위로 묶어 넣고 experss.Router()의 객체를 사용해야 한다는 차이점이 있습니다. Router-level 미들웨어를 사용한다면 어플리케이션에 Path별 요청에 따른 동작 방식을 모듈화하여 관리할 수 있습니다.

## Error Handling

- 에러 처리를 담당하는 미들웨어입니다. 이러한 유형의 미들웨어는 반드시 네 개의 인자를 매개변수로 받아 이 미들웨어가 에러를 담당하는 미들웨어라는 것을 식별해야 합니다.

## Third-party

- 기본적으로 주어지는 Built-in middleware 외에 추가로 설치하여 사용해야하는 미들웨어를 Third-party middleware라고 합니다.

# 미들웨어의 유형

## 1. Application-level middleware

```
app.use(function(req, res, next){
  console.log('Time: ', Date.now())
  next()
}) // ----- 1
app.use('/index', function(req, res, next){
  /*
      Code
  */
}) // ----- 2
```

- app.use() 를 사용할 때는 path를 지정할 수 있습니다.
- 첫 번째 미들웨어는 어떤 request가 들어와도 작동을 하는 것이고
- 두 번째 미들웨어는 /index의 path를 통해 들어오는 요청에 대해서만 작동하는 미들웨어입니다.

# 미들웨어의 유형
## 4. Third-party 미들웨어 : body-parser

- 요청의 본문을 해석해주는 미들웨어
  - 폼 데이터나 AJAX 요청의 데이터 처리
  - 요청에 대한 처리 속성 : request.body
- 설치
  - $ npm install body-parser --save
- API
  - var bodyParser = require('body-parser')
- parse application/x-www-form-urlencoded
  - app.use(bodyParser.urlencoded({ extended: false }))

# 4. Third-party 미들웨어 : compression

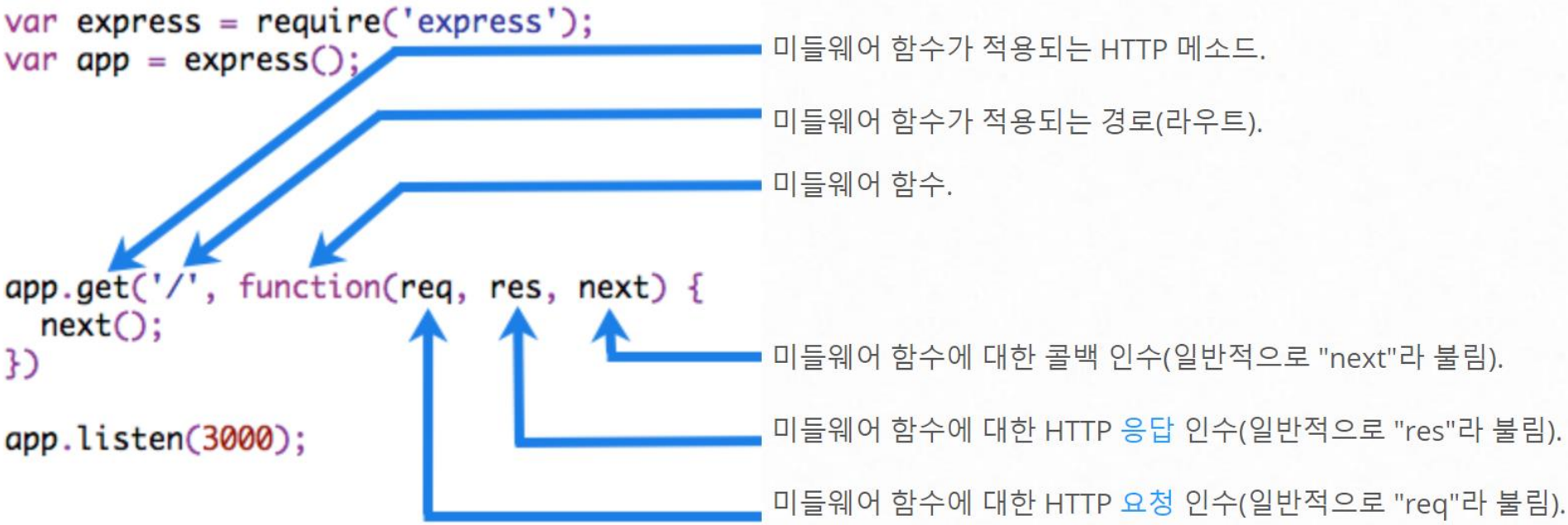


- compression 미들웨어를 이용해서 컨텐츠를 압축해서 전송하는 방법
- Install
  - $ npm install compression --save
- API
  - var compression = require('compression')
- Examples
  - app.use(compression())

# Express 미들웨어 만들기

Node.js 교과서

- 미들웨어 함수는
  - 요청 오브젝트(req), 응답 오브젝트 (res),
  - 그리고 애플리케이션의 요청-응답 주기 중 그 다음의 미들웨어 함수 대한 액세스 권한을 갖는 함수입니다.
  - 그 다음의 미들웨어 함수는 일반적으로 next라는 이름의 변수로 표시됩니다.
  - 따라서, next()함수는 호출한 곳 다음의 프로세스를 진행하는 의미



# 익스프레스에서 정적인 파일의 서비스


- 이미지, 자바스크립트, CSS와 같은 파일을 서비스하는 방법
- 이미지, CSS 파일 및 JavaScript 파일과 같은 정적 파일을 제공하려면 Express의 기본 제공 미들웨어 함수인 express.static을 사용
- 정적 자산이 포함된 디렉토리의 이름을 express.static 미들웨어 함수에 전달하면 파일의 직접적인 제공을 시작
- 다음과 같은 코드를 이용하여 public이라는 이름의 디렉토리에 포함된 이미지, CSS 파일 및 JavaScript 파일을 제공
  - app.use(express.static('public'));
  - 이제 다음과 같이 public 디렉토리에 포함된 파일을 로드할 수 있습니다.
  - http://localhost:3000/images/kitten.jpg
  - http://localhost:3000/css/style.css
  - http://localhost:3000/js/app.js
  - http://localhost:3000/images/bg.png
  - http://localhost:3000/hello.html

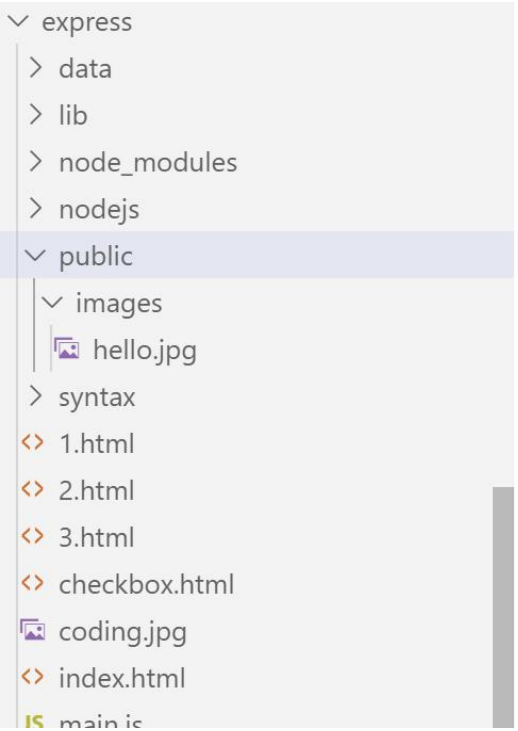
# 미들웨어 함수 express.static 사용

Node.js 교과서

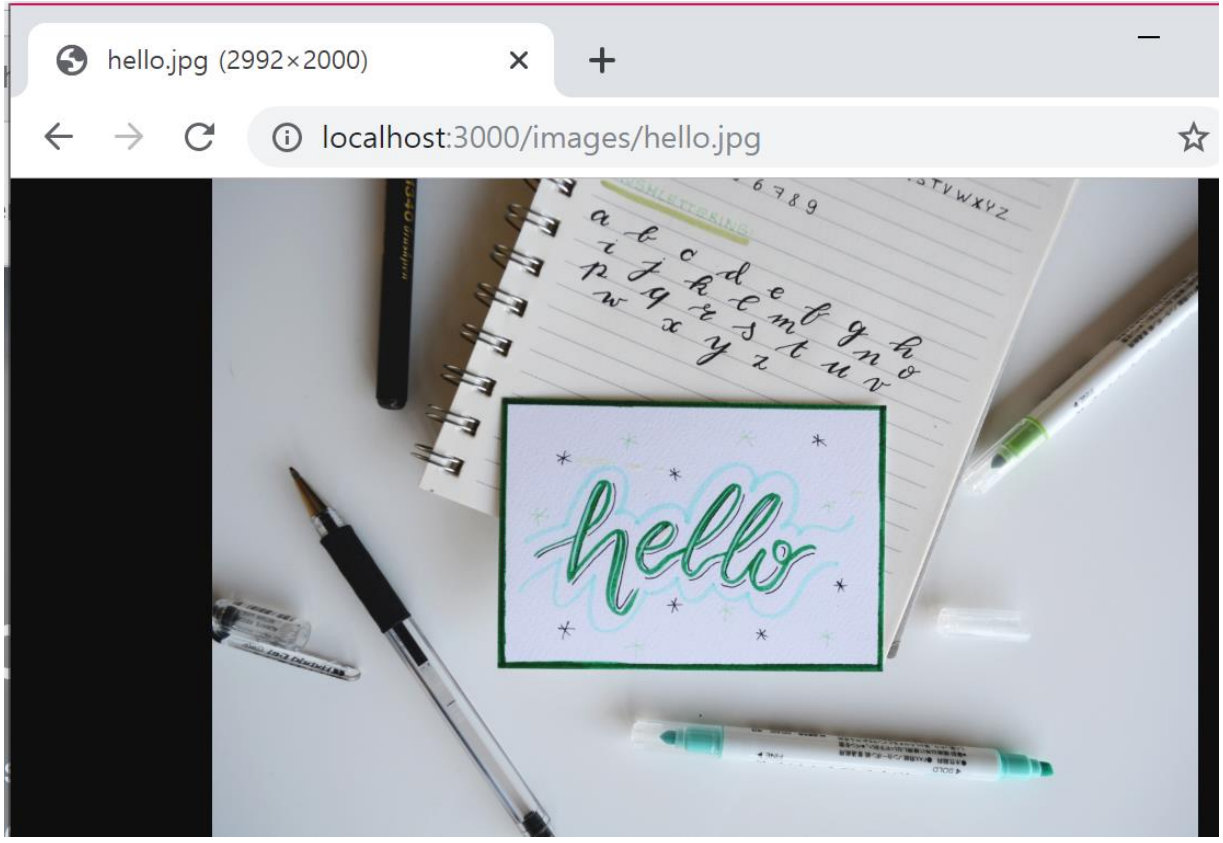
```
1   var express = require('express')
2   var app = express()
3   var fs = require('fs');
4   var template = require('./lib/template.js');
5   var path = require('path');
6   var qs = require('querystring');
7   var sanitizeHtml = require('sanitize-html');
8   var bodyParser = require('body-parser');
9   var compression = require('compression')
10
11  app.use(express.static('public'));
12  app.use(bodyParser.urlencoded({ extended: false }));
13  app.use(compression());
```

# 익스프레스에서 정적인 파일의 서비스

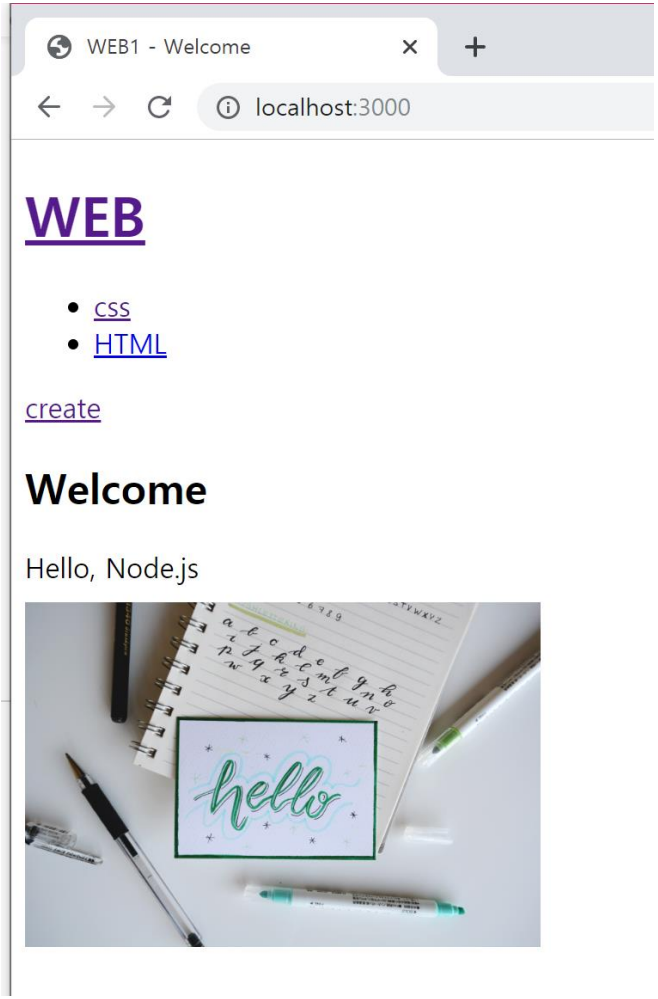
- 이미지 파일 다운로드 후 express > public > images 폴더를 생성해서 저장
- https://unsplash.com/




- 다음과 같은 URL 입력하여 public이라는 이름의 디렉토리에 포함된 이미지가 제공되는지 확인
- 미들웨어 함수인 express.static을 사용해 제공되는 이미지 경로를 확인 "/images/hello.jpg"



## 메인페이지에 이미지 추가해 보기



```
21  app.get('/', function(request, response) {
22      var title = 'Welcome';
23      var description = 'Hello, Node.js';
24      var list = template.list(request.list);
25      var html = template.HTML(title, list,
26        `<h2>${title}</h2>${description}
27        <img src="/images/hello.jpg" style="width:300px;
28        display:block; margin-top:10px;">`,
29        `<a href="/create">create</a>`
30      );
31      response.send(html);
32  });
```

# 익스프레스 에러처리



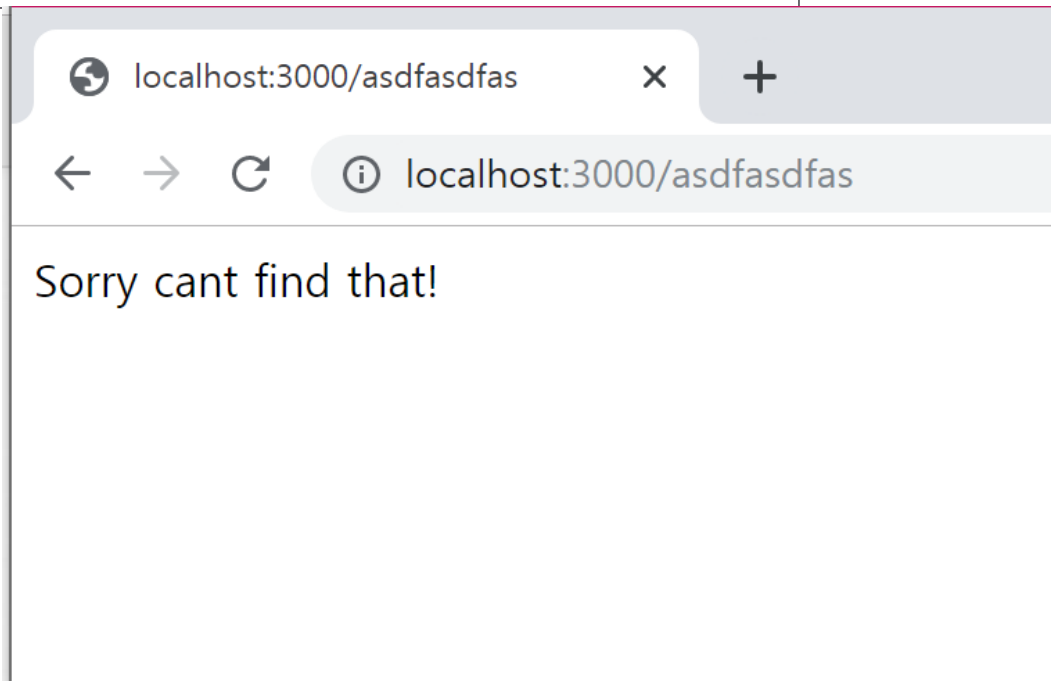
## Express에서 404 응답

- 404 응답은 단순히 실행해야 할 추가적인 작업이 없다는 에러
- 즉 Express는 모든 미들웨어 함수 및 라우트를 실행했으며 이들 중 어느 것도 응답하지 않았다는 것을 나타내기 때문
- 이를 처리하려면 다음과 같이 404 응답을 처리하기 위한 미들웨어 함수를 스택의 가장 아래(다른 모든 함수의 아래)에 추가

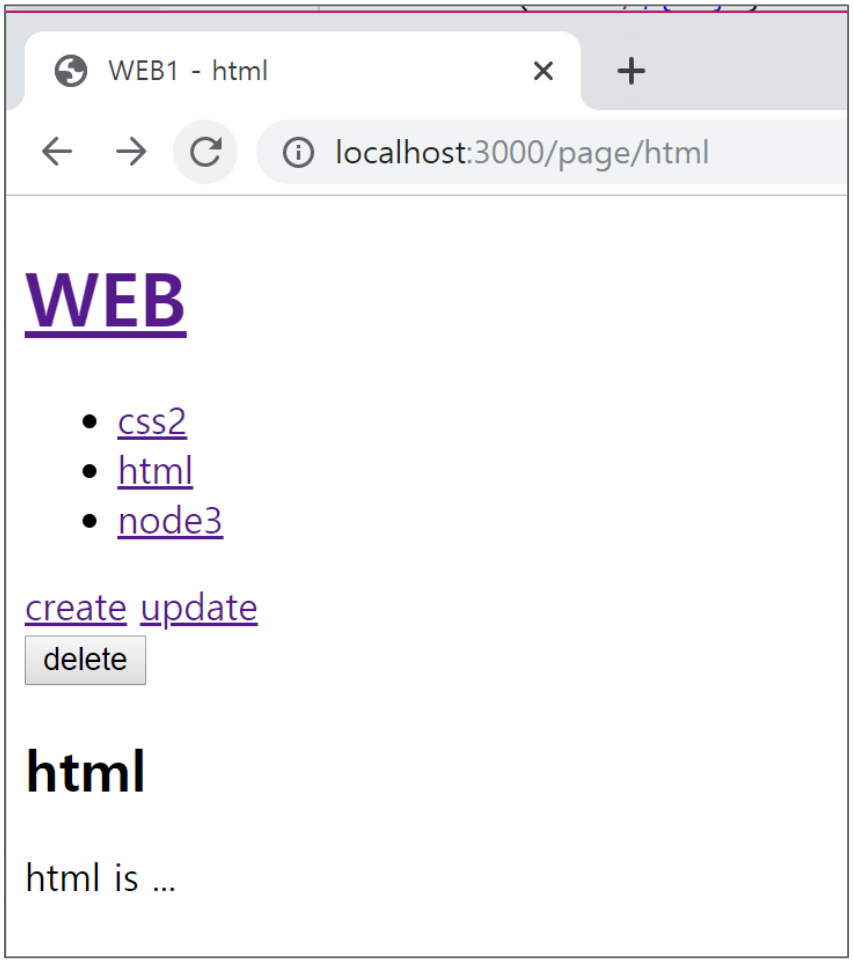
```
app.use(function(req, res, next) {
   res.status(404).send('Sorry cant find that!');
});
```

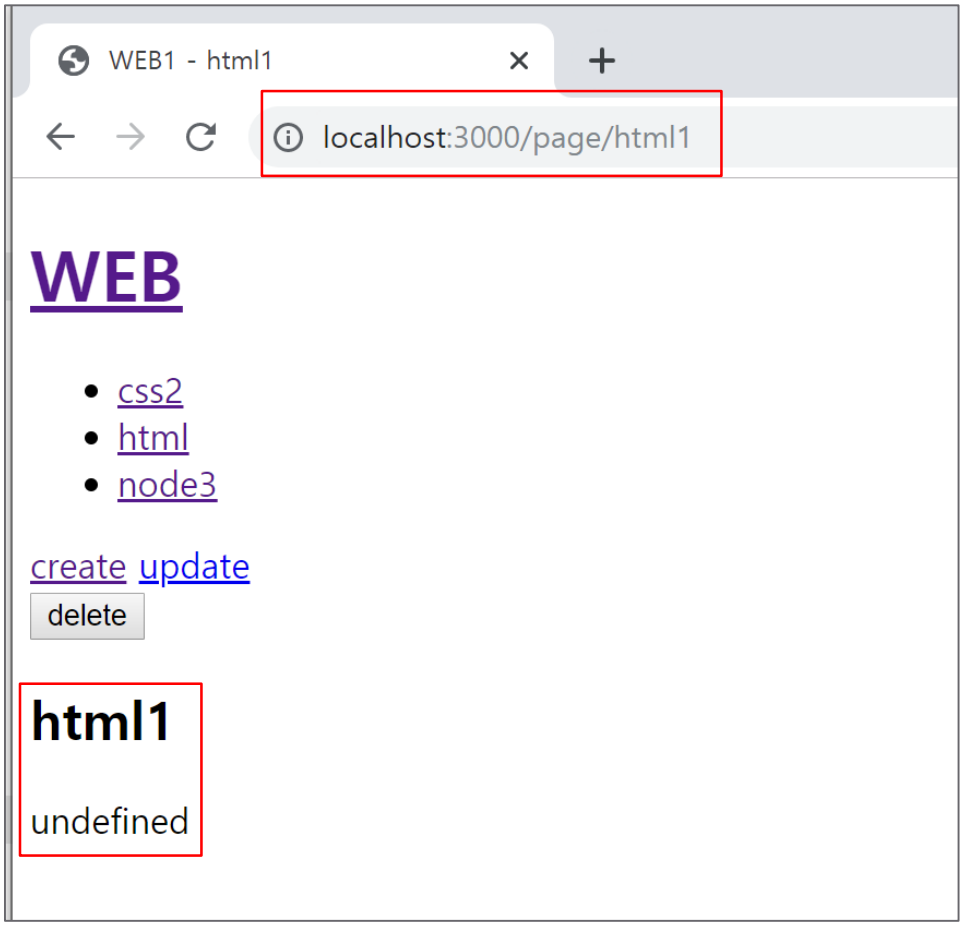
# 익스프레스 에러처리

```
127  app.use(function(req, res, next) {
128    res.status(404).send('Sorry cant find that!');
129  });
130
131  app.listen(3000, function() {
132    console.log('Example app listening on port 3000!')
133  });
```



# pageId 입력 오류 처리



# pageId 입력 오류 처리
# - 오류 처리기 작성


- '500'에러 처리
- 기존 상세페이지
  - 라우팅 처리 콜백함수에 'next'인수를 추가
  - fs.readFile 함수의 콜백함수 err 인수에 대한 오류 메시지 처리
- 오류 처리 함수정의
  - 함수 인수에 세 개가 아닌 네 개의 인수가 있다는 점을 제외하고 다른 미들웨어 함수와 같은 방식으로 오류 처리 미들웨어 함수를 정의

```
app.use(function (err, req, res, next) {
  console.error(err.stack)
  res.status(500).send('Something broke!')
})
```

# 기존 상세페이지 처리 코드

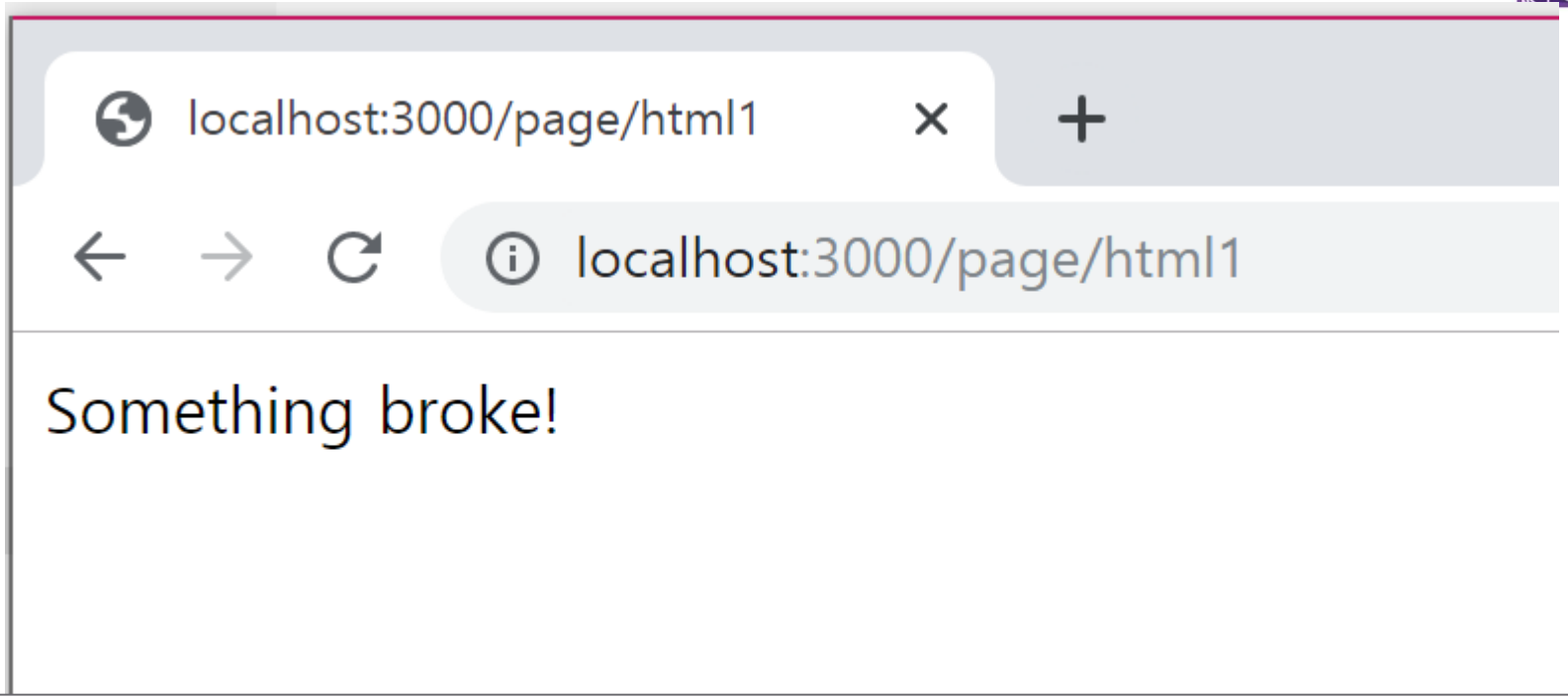
Node.js 교과서

```
34  app.get('/page/:pageId', function(request, response) {
35    var filteredId = path.parse(request.params.pageId).base;
36    fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
37      var title = request.params.pageId;
38      var sanitizedTitle = sanitizeHtml(title);
39      var sanitizedDescription = sanitizeHtml(description, {
40        allowedTags:['h1']
41      });
42      var list = template.list(request.list);
43      var html = template.HTML(sanitizedTitle, list,
44        `<h2>${sanitizedTitle}</h2>${sanitizedDescription}`,
45        ` <a href="/create">create</a>
46          <a href="/update/${sanitizedTitle}">update</a>
47          <form action="/delete_process" method="post">
48            <input type="hidden" name="id" value="${sanitizedTitle}">
49            <input type="submit" value="delete">
50          </form>`
51      );
52      response.send(html);
53    });
54  });
```

```
34  app.get('/page/:pageId', function(request, response, next) {
35    var filteredId = path.parse(request.params.pageId).base;
36    fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
37      if(err){
38        next(err);
39      } else{
40        var title = request.params.pageId;
41        var sanitizedTitle = sanitizeHtml(title);
42        var sanitizedDescription = sanitizeHtml(description, {
43          allowedTags:['h1']
44        });
45        var list = template.list(request.list);
46        var html = template.HTML(sanitizedTitle, list,
47          `<h2>${sanitizedTitle}</h2>${sanitizedDescription}`,
48          ` <a href="/create">create</a>
49            <a href="/update/${sanitizedTitle}">update</a>
50            <form action="/delete_process" method="post">
51              <input type="hidden" name="id" value="${sanitizedTitle}">
52              <input type="submit" value="delete">
53            </form>`
54        );
55        response.send(html);
56      }
57    });
58  });
```

# 오류 처리 함수정의


```
131   app.use(function(req, res, next) {
132     res.status(404).send('Sorry cant find that!');
133   });
134
135   app.use(function (err, req, res, next) {
136     console.error(err.stack)
137     res.status(500).send('Something broke!')
138   });
139
140   app.listen(3000, function() {
141     console.log('Example app listening on port 3000!')
142   });
```




```
PS C:\express\express> node main.js
Example app listening on port 3000!
PS C:\express\express> node main.js
Example app listening on port 3000!
Error: ENOENT: no such file or directory, open 'C:\express\express\data\html1'
```

# express.Router 클래스

- express.Router클래스 사용위해 마운트 가능한 모듈식 라우트 핸들러를 작성
- Router인스턴스는 완전한 미들웨어 및 라우팅 시스템으로 종종 "미니 앱"이라고 함
- 다음 예제는 라우터를 모듈로 작성하고 미들웨어 기능을로드하고 일부 라우트를 정의한 후 라우터 모듈을 기본 앱의 경로에 마운트합니다.

```
var birds = require('./birds')

// ...

app.use('/birds', birds)
```

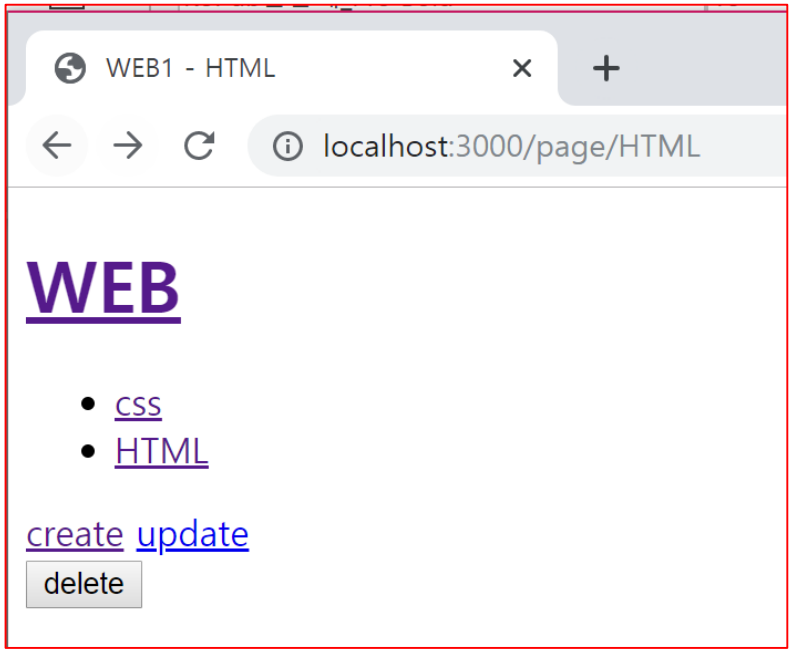
```
var express = require('express')
var router = express.Router()

// middleware that is specific to this router
router.use(function timeLog (req, res, next) {
  console.log('Time: ', Date.now())
  next()
})
// define the home page route
router.get('/', function (req, res) {
  res.send('Birds home page')
})
// define the about route
router.get('/about', function (req, res) {
  res.send('About birds')
})

module.exports = router
```

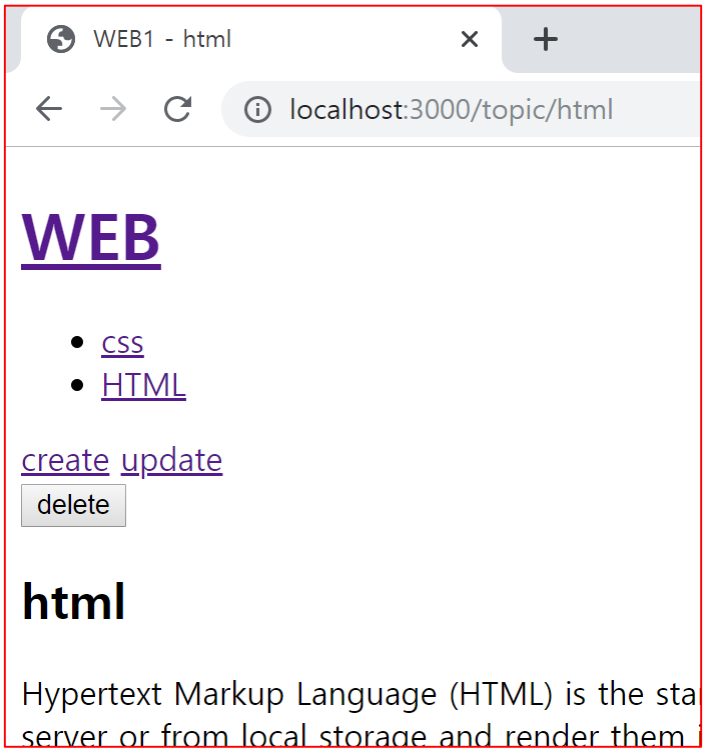
/birds 에 대한 요청 처리

/birds/about 에 대한 요청 처리

# express.Router사용-라우터 정리

- 기존 앱의 주소체계를 변경하는 작업
- /page/HTML -> /topic/HTML
- 라우팅 순서에 주의



# 상세페이지 수정

Node.js 교과서

```
app.get('/page/:pageId', function(requ
    var filteredId = path.parse(reques
    fs.readFile(`data/${filteredId}`,
      if(err){
        next(err);
      } else{
        var title = request.params.pag
        var sanitizedTitle = sanitizeH
        var sanitizedDescription = san
          allowedTags:['h1']
        });
        var list = template.list(reque
        var html = template.HTML(sanit
          `<h2>${sanitizedTitle}</h2>$
          ` <a href="/create">create</
            <a href="/update/${sanitiz
            <form action="/delete_proc
              <input type="hidden" nam
              <input type="submit" val
            </form>`
        );
        response.send(html);
      }
    });
});
```

```
app.get('/topic/:pageId', function(request, response, next) {
    var filteredId = path.parse(request.params.pageId).base;
    fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
      if(err){
        next(err);
      } else{
        var title = request.params.pageId;
        var sanitizedTitle = sanitizeHtml(title);
        var sanitizedDescription = sanitizeHtml(description, {
          allowedTags:['h1']
        });
        var list = template.list(request.list);
        var html = template.HTML(sanitizedTitle, list,
          `<h2>${sanitizedTitle}</h2>${sanitizedDescription}`,
          ` <a href="/create">create</a>
            <a href="/update/${sanitizedTitle}">update</a>
            <form action="/delete_process" method="post">
              <input type="hidden" name="id" value="${sanitizedTitle}">
              <input type="submit" value="delete">
            </form>`
        );
        response.send(html);
      }
    });
});
```

express > lib > JS template.js > ...

```
1   module.exports = {
2 >   HTML:function(title, list, body, control){ ···
18    },list:function(filelist){
19      var list = '<ul>';
20      var i = 0;
21      while(i < filelist.length){
22        list = list + `<li><a href="/topic/${filelist[i]}">${filelist[i]}</a></li>`;
23        i = i + 1;
24      }
25      list = list+'</ul>';
26      return list;
27    }
28  }
```

express > lib > JS template.js > <unknown> > list

```
1   module.exports = {
2 >   HTML:function(title, list, body, control){ ···
18    },list:function(filelist){
19      var list = '<ul>';
20      var i = 0;
21      while(i < filelist.length){
22        list = list + `<li><a href="/page/${filelist[i]}">${filelist[i]}</a></li>`;
23        i = i + 1;
24      }
25      list = list+'</ul>';
26      return list;
27    }
28  }
```

# /create 수정

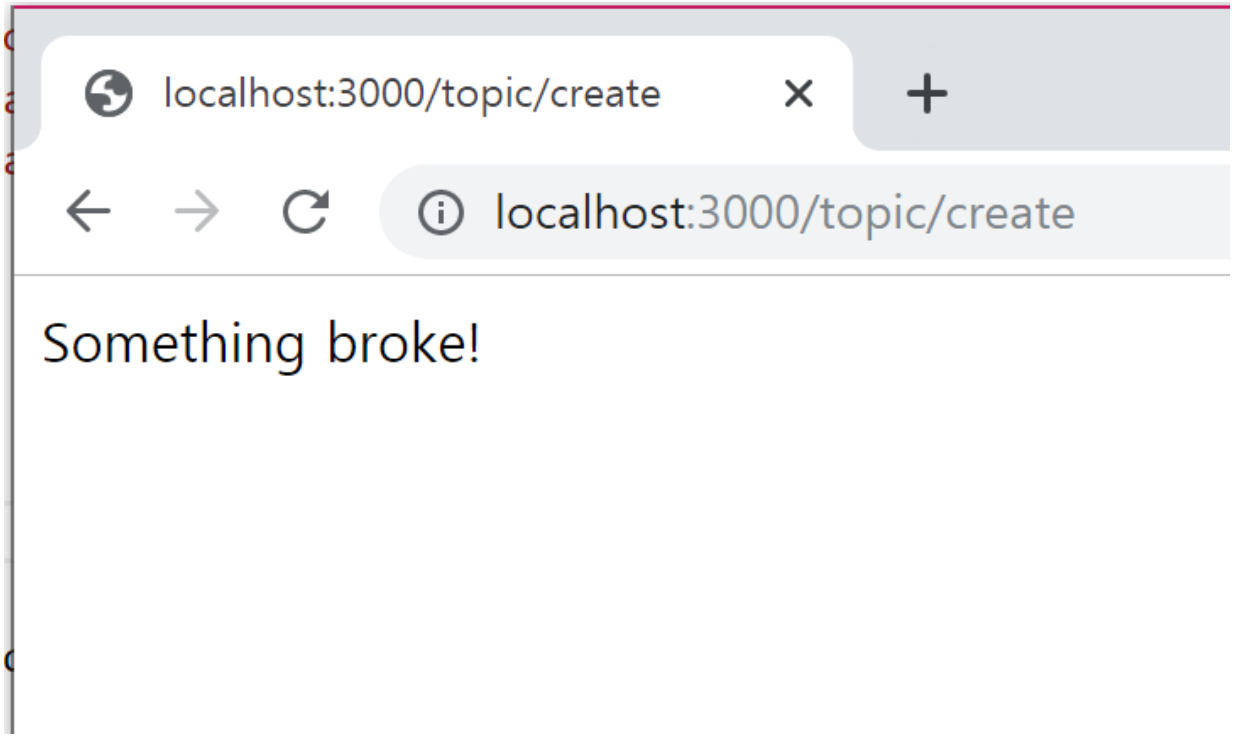

```
21  app.get('/', function(request, response) {
22      var title = 'Welcome';
23      var description = 'Hello, Node.js';
24      var list = template.list(request.list);
25      var html = template.HTML(title, list,
26        `<h2>${title}</h2>${description}
27        <img src="/images/hello.jpg" style="width:300px;
28        display:block; margin-top:10px;">`,
29        `<a href="/topic/create">create</a>`
30      );
31      response.send(html);
32  });
```

# /create 수정

```
34  app.get('/topic/:pageId', function(request, response, next) {
35      var filteredId = path.parse(request.params.pageId).base;
36      fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
37        if(err){
38          next(err);
39        } else{
40          var title = request.params.pageId;
41          var sanitizedTitle = sanitizeHtml(title);
42          var sanitizedDescription = sanitizeHtml(description, {
43            allowedTags:['h1']
44          });
45          var list = template.list(request.list);
46          var html = template.HTML(sanitizedTitle, list,
47            `<h2>${sanitizedTitle}</h2>${sanitizedDescription}`,
48            ` <a href="/topic/create">create</a>
49              <a href="/update/${sanitizedTitle}">update</a>
50              <form action="/delete_process" method="post">
51                <input type="hidden" name="id" value="${sanitizedTitle}">
52                <input type="submit" value="delete">
53              </form>`
54          );
55          response.send(html);
56        }
57      });
58  });
59
60  app.get('/topic/create', function(request, response){···
75  });
```

# /create 라우팅 오류처리

- url 주소에 대한 분석을 해당 폴더의 파일을 찾는 것으로 처리함
- 그런데  create 라는 것을 파일을 의미하는 것이 아니고 특별한 처리를 의미하는 것으로 라우팅 순서를 변경하면 해결할 수 있음



# /create 라우팅 오류처리
# -기존 라우팅 순서


```
14 > app.get('*',function(request, response, next){ ···
19   });
20
21 > app.get('/', function(request, response) { ···
32   });
33
34 > app.get('/topic/:pageId', function(request, response, next) { ···
58   });
59
60 > app.get('/topic/create', function(request, response){ ···
75   });
76
77 > app.post('/topic/create_process', function(request, response){ ···
84   });
```

# /create 라우팅 오류처리
# - 라우팅 순서 변경


```
14 > app.get('*',function(request, response, next){ ···
19   });
20
21 > app.get('/', function(request, response) { ···
32   });
33
34 > app.get('/topic/create', function(request, response){ ···
49   });
50
51 > app.post('/topic/create_process', function(request, response){ ···
58   });
59
60 > app.get('/topic/:pageId', function(request, response, next) { ···
84   });
```

# /create_process 수정


```
34  app.get('/topic/create', function(request, response){
35    var title = 'WEB - create';
36    var list = template.list(request.list);
37    var html = template.HTML(title, list, `
38      <form action="/topic/create_process" method="post">
39        <p><input type="text" name="title" placeholder="title"></p>
40        <p>
41          <textarea name="description" placeholder="description"></textarea>
42        </p>
43        <p>
44          <input type="submit">
45        </p>
46      </form>
47    `, '');
48    response.send(html);
49  });
```

# /create_process 수정


```
51  app.post('/topic/create_process', function(request, response){
52    var post = request.body;
53    var title = post.title;
54    var description = post.description;
55    fs.writeFile(`data/${title}`, description, 'utf8', function(err){
56      response.redirect(`/topic/${title}`);
57    });
58  });
```

# /update 수정

```
96  app.get('/topic/:pageId', function(request, response, next) {
97      var filteredId = path.parse(request.params.pageId).base;
98      fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
99        if(err){
100         next(err);
101       } else{
102         var title = request.params.pageId;
103         var sanitizedTitle = sanitizeHtml(title);
104         var sanitizedDescription = sanitizeHtml(description, {
105           allowedTags:['h1']
106         });
107         var list = template.list(request.list);
108         var html = template.HTML(sanitizedTitle, list,
109           `<h2>${sanitizedTitle}</h2>${sanitizedDescription}`,
110           ` <a href="/topic/create">create</a>
111             <a href="/topic/update/${sanitizedTitle}">update</a>
112             <form action="/delete_process" method="post">
113               <input type="hidden" name="id" value="${sanitizedTitle}">
114               <input type="submit" value="delete">
115             </form>`
116         );
117         response.send(html);
118       }
119     });
120 });
```

# /update 라우팅 순서 변경

```
14 > app.get('*',function(request, response, next){ ···
19   });
20
21 > app.get('/', function(request, response) { ···
32   });
33
34 > app.get('/topic/create', function(request, response){ ···
49   });
50
51 > app.post('/topic/create_process', function(request, response){ ···
58   });
59
60 > app.get('/topic/update/:pageId', function(request, response){ ···
82   });
83
84 > app.post('/topic/update_process', function(request, response){ ···
94   });
95
96 > app.get('/topic/:pageId', function(request, response, next) { ···
20   });
21
```

# /update_process 수정

Node.js 교과서

```
60 app.get('/topic/update/:pageId', function(request, response){
61   var filteredId = path.parse(request.params.pageId).base;
62   fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
63     var title = request.params.pageId;
64     var list = template.list(request.list);
65     var html = template.HTML(title, list,
66       `
67       <form action="/topic/update_process" method="post">
68         <input type="hidden" name="id" value="${title}">
69         <p><input type="text" name="title" placeholder="title" value="${title}"></p>
70         <p>
71           <textarea name="description" placeholder="description">${description}</textarea>
72         </p>
73         <p>
74           <input type="submit">
75         </p>
76       </form>
77       `,
78       `<a href="/topic/create">create</a> <a href="/topic/update/${title}">update</a>`
79     );
80     response.send(html);
81   });
82 });
```

# /update_process 수정


```
84  app.post('/topic/update_process', function(request, response){
85    var post = request.body;
86    var id = post.id;
87    var title = post.title;
88    var description = post.description;
89    fs.rename(`data/${id}`, `data/${title}`, function(error){
90      fs.writeFile(`data/${title}`, description, 'utf8', function(err){
91        response.redirect(`/topic/${title}`);
92      })
93    });
94  });
```

# /delete 수정 및 라우팅 순서 변경

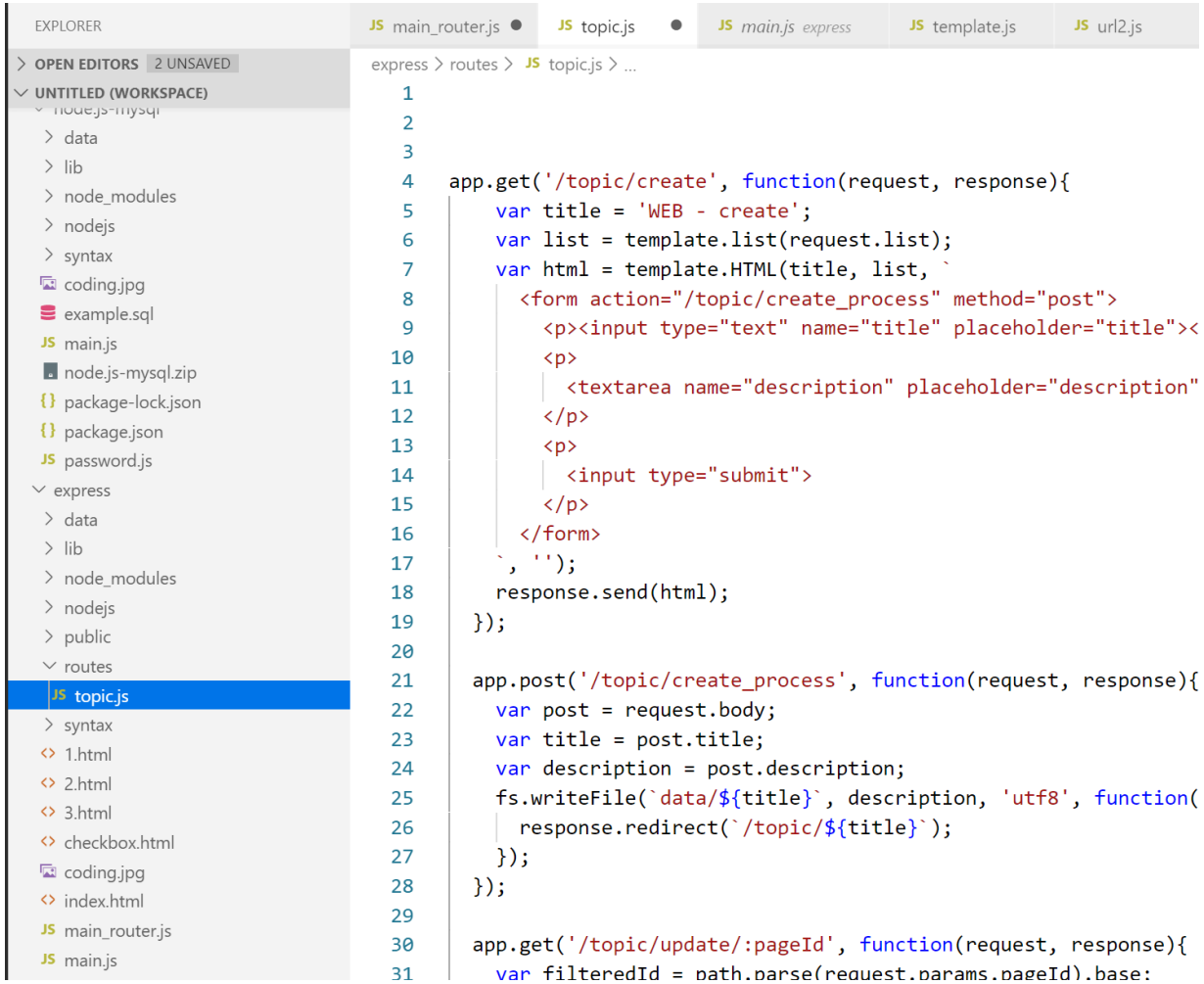
```
96  app.post('/topic/delete_process', function(request, response){
97    var post = request.body;
98    var id = post.id;
99    var filteredId = path.parse(id).base;
100   fs.unlink(`data/${filteredId}`, function(error){
101     response.redirect('/');
102   });
103 });
104
105 app.get('/topic/:pageId', function(request, response, next) {
106     var filteredId = path.parse(request.params.pageId).base;
107     fs.readFile(`data/${filteredId}`, 'utf8', function(err, description){
108       if(err){
109         next(err);
110       } else{
111         var title = request.params.pageId;
112         var sanitizedTitle = sanitizeHtml(title);
113         var sanitizedDescription = sanitizeHtml(description, {
114           allowedTags:['h1']
115         });
116         var list = template.list(request.list);
117         var html = template.HTML(sanitizedTitle, list,
118           `<h2>${sanitizedTitle}</h2>${sanitizedDescription}`,
119           ` <a href="/topic/create">create</a>
120             <a href="/topic/update/${sanitizedTitle}">update</a>
121             <form action="/topic/delete_process" method="post">
122               <input type="hidden" name="id" value="${sanitizedTitle}">
123               <input type="submit" value="delete">
124             </form>`
125         );
126         response.send(html);
127       }
128     });
129 });
```

# /delete 수정 및 라우팅 순서 변경

```
app.get('/', function(request, response) { ···
});

app.get('/topic/create', function(request, response){···
});

app.post('/topic/create_process', function(request, response){···
});

app.get('/topic/update/:pageId', function(request, response){···
});

app.post('/topic/update_process', function(request, response){···
});

app.post('/topic/delete_process', function(request, response){···
});

app.get('/topic/:pageId', function(request, response, next) { ···
});
```

# 라우터를 만들어서 파일로 분리

- 작업폴더에 routes > topic.js 생성
- main 파일에서 'topic'으로 시작하는 라우팅 모듈을 가져와서 topic.js 저장



# 라우터를 만들어서 파일로 분리

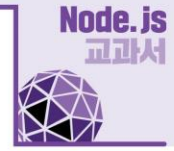


» main 파일에서 'topic'으로 시작하는 라우팅 모듈을 가져와서 topic.js 저장

express > routes > JS topic.js > ...

```
1
2
3
4  > router.get('/topic/create', function(request, response){ ···
19   });
20
21 > router.post('/topic/create_process', function(request, response){ ···
28   });
29
30 > router.get('/topic/update/:pageId', function(request, response){ ···
52   });
53
54 > router.post('/topic/update_process', function(request, response){ ···
64   });
65
66 > router.post('/topic/delete_process', function(request, response){ ···
73   });
74
75 > router.get('/topic/:pageId', function(request, response, next) { ···
99   });
100
```

# Main 파일에 라우터모듈 처리

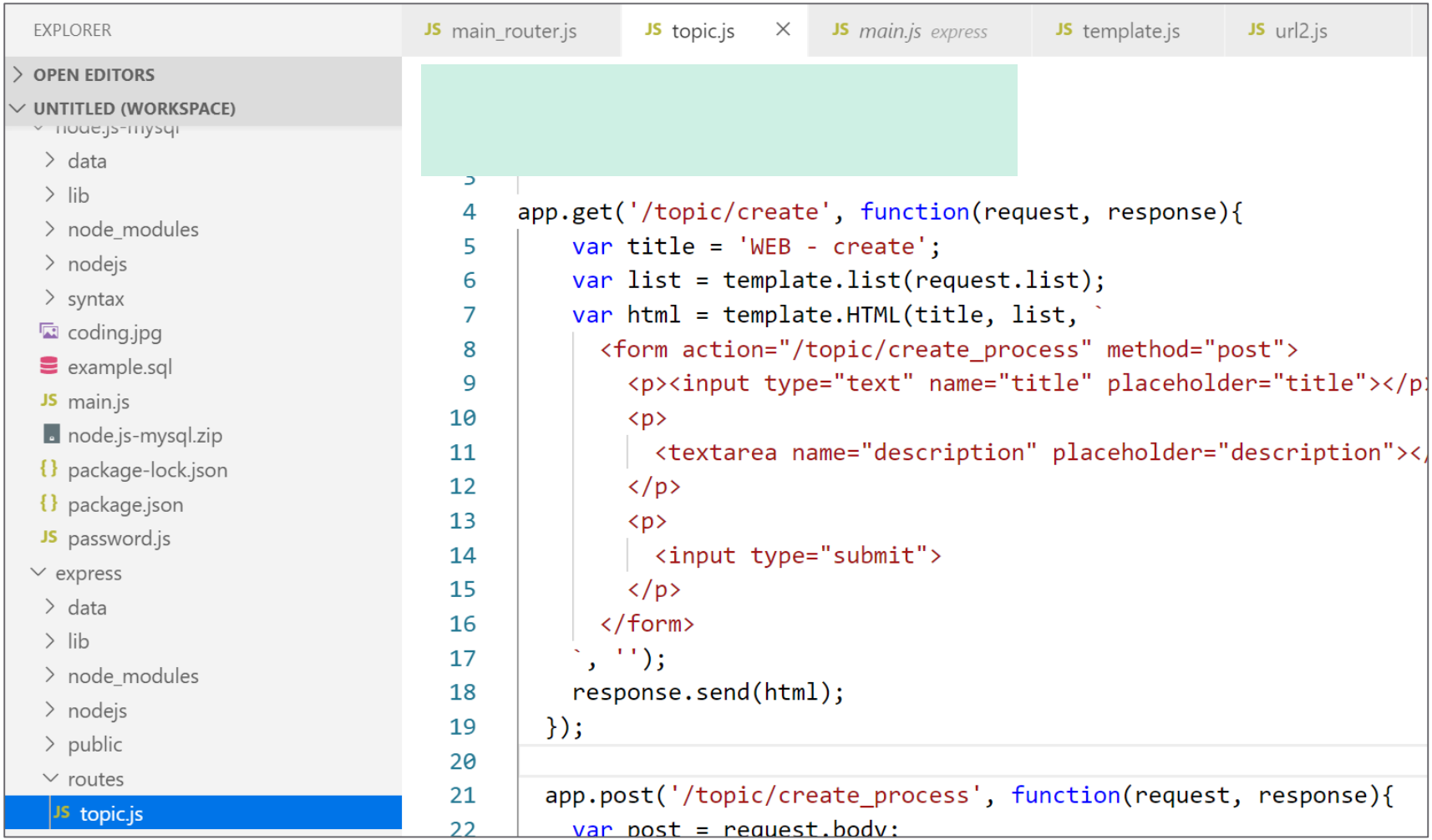



express > JS main_router.js > ...

```
1   var express = require('express')
2   var app = express()
3   var fs = require('fs');
4   var template = require('./lib/template.js');
5   var path = require('path');
6   var qs = require('querystring');
7   var sanitizeHtml = require('sanitize-html');
8   var bodyParser = require('body-parser');
9   var compression = require('compression');
```

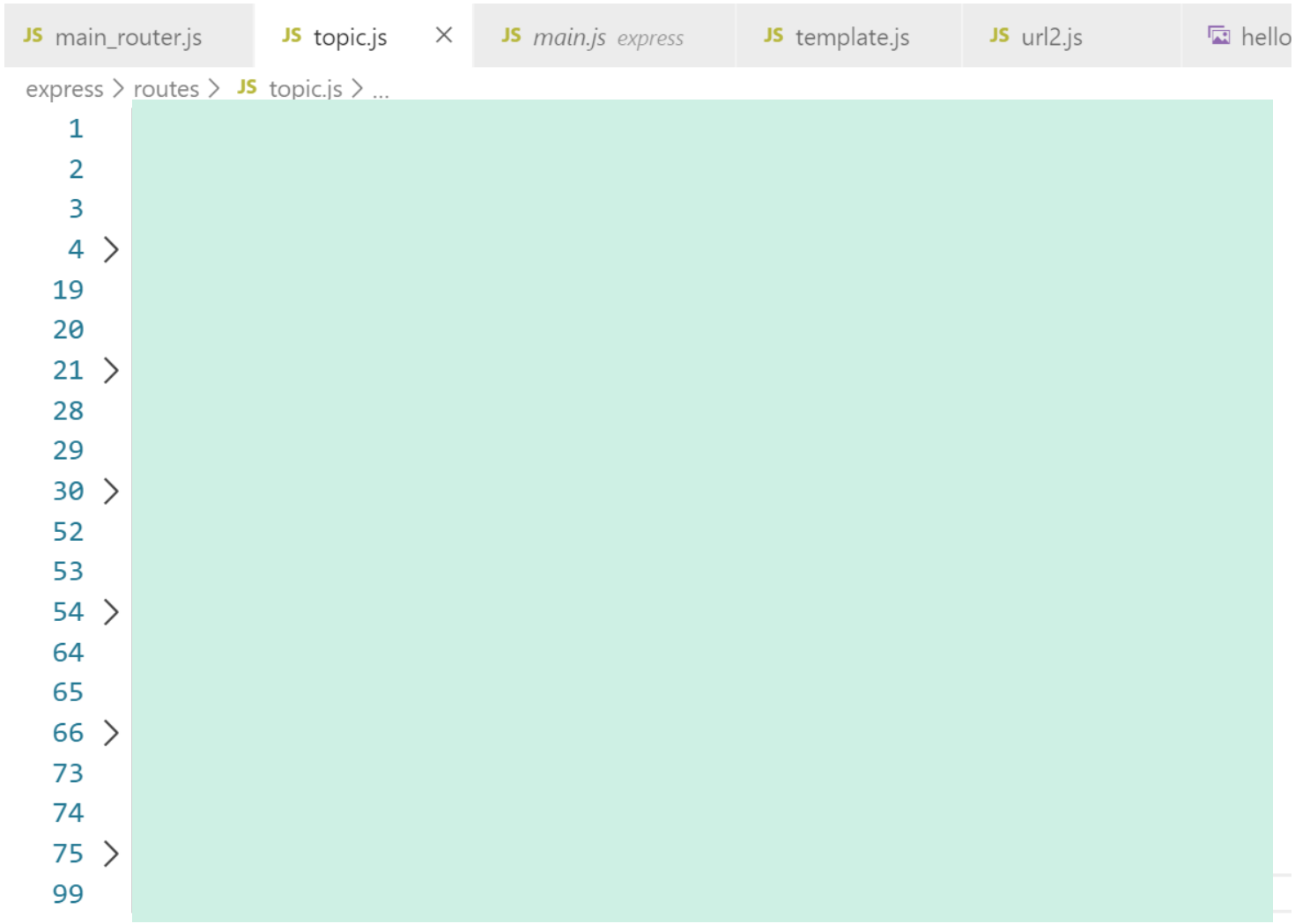
# topic.js 수정

- topic.js 에서 express.Router()모듈 선언



# topic.js 수정

- topic.js 에서 기존 app 변수를 router 변수로 변경



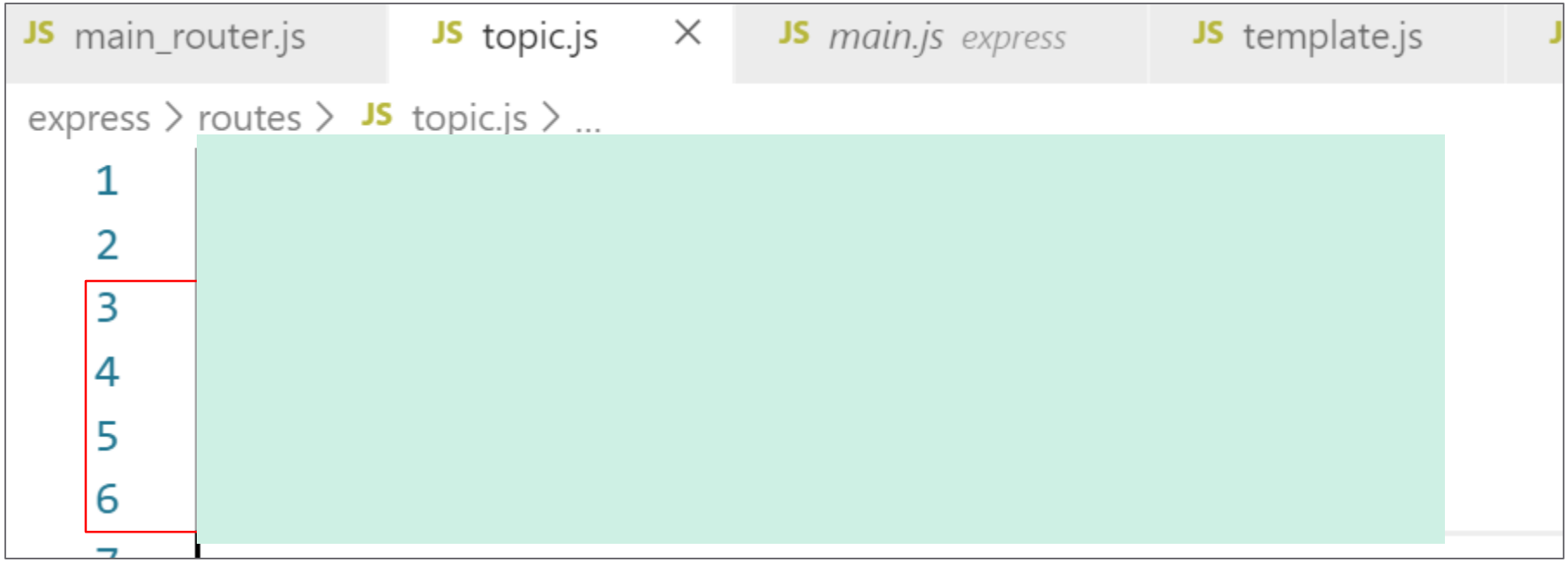
# topic.js 수정

- 라우팅 경로에서 '/topic'삭제(**express.Router 라우팅 특성)**
- router 모듈 exports 선언하기

# topic.js 수정

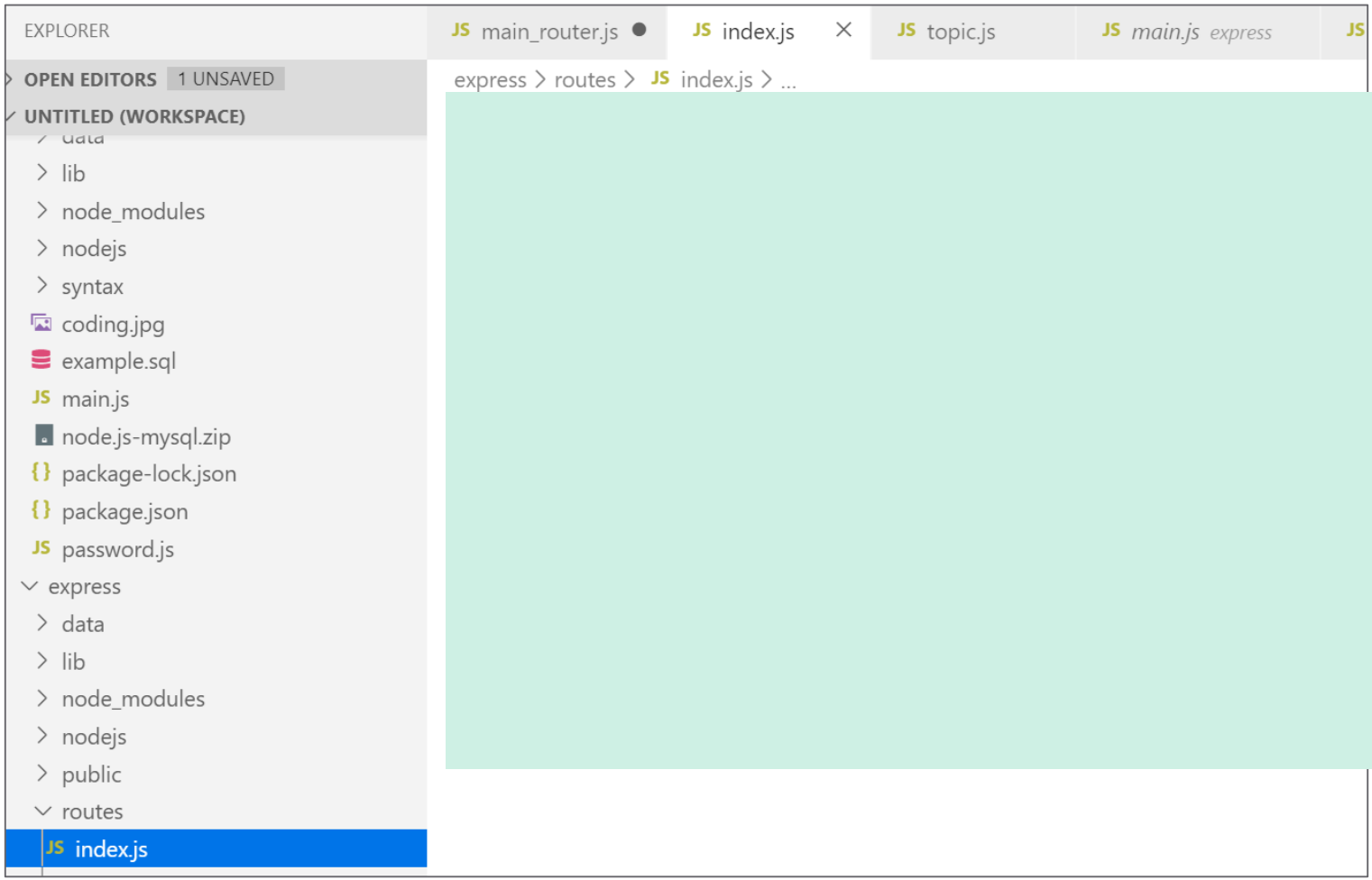


- 필요한 모듈 선언하기



# 홈페이지를 파일로 분리하기

- 작업폴더에 routes > index.js 생성
- main 파일에서 '/'으로 시작하는 라우팅 모듈을 가져와서 index.js 저장



## index.js 수정

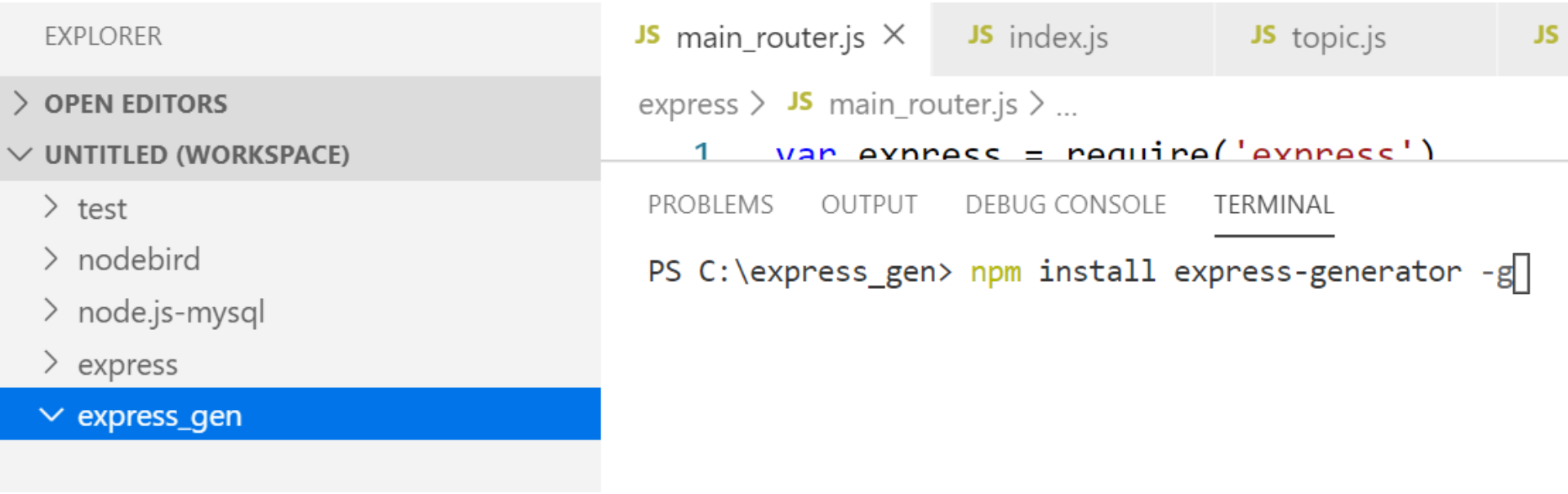
express > routes > JS index.js > ...

» main.js 수정

```
1
2
3
4
5
6
7
8
9
10
11
12
17
18
19
20
21
```

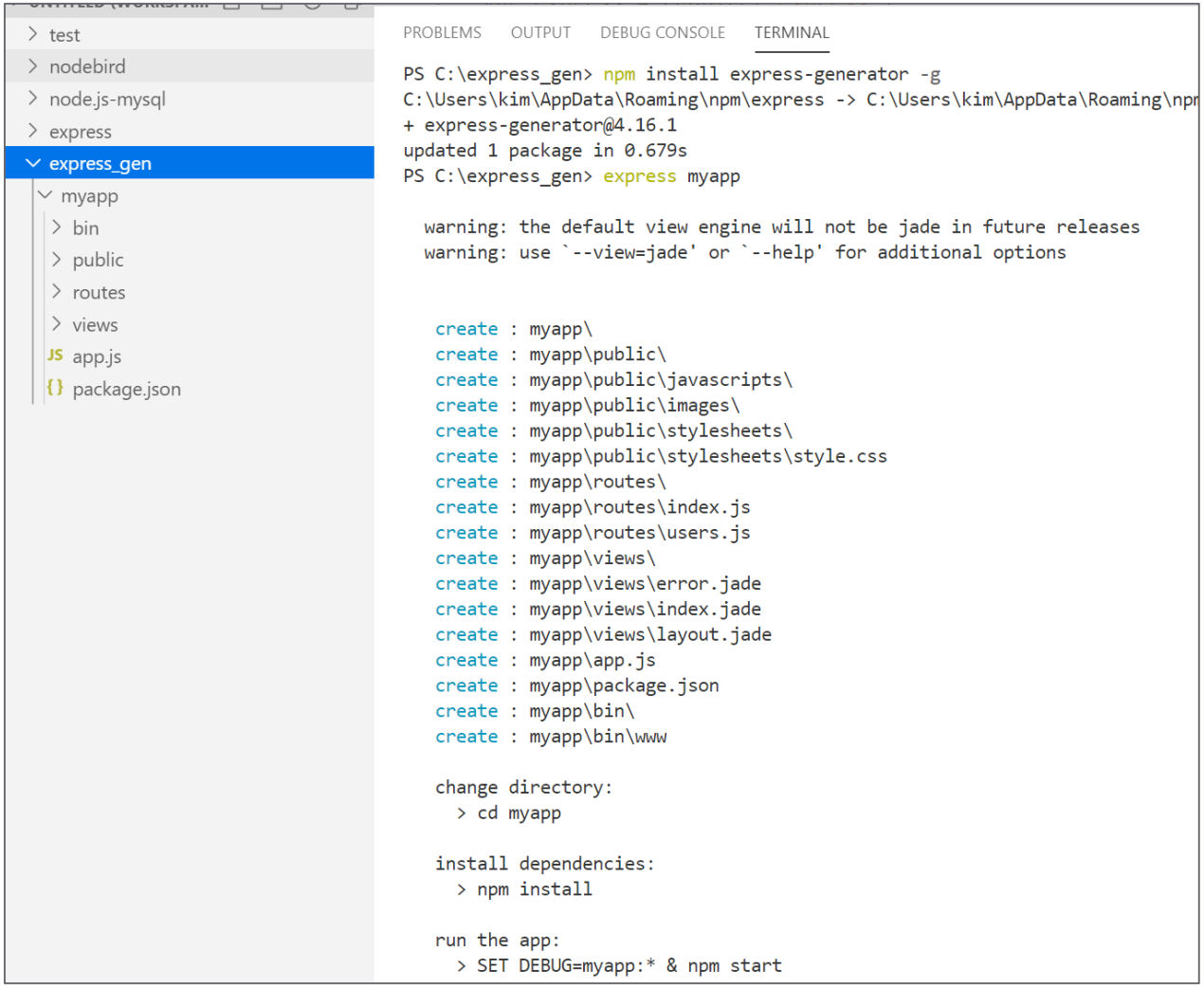
# Express generator

- Express 기반의 프로젝트를 할 때 기본적으로 필요한 파일과 코드를 자동으로 만들어주는 앱인
- 다음의 명령을 새로운 폴더에서 실행해 express를 설치
- $ npm install express-generator -g



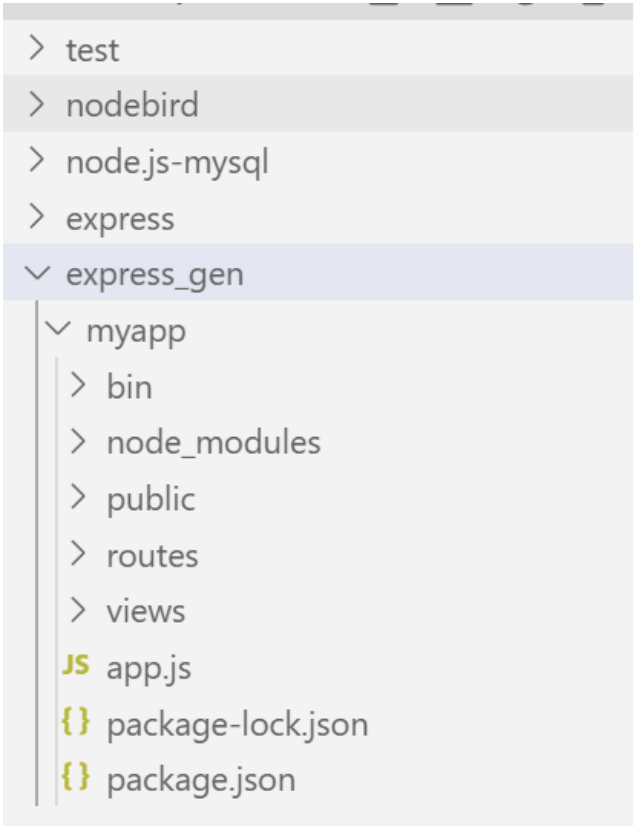
# 익스프레스 프로젝트 생성

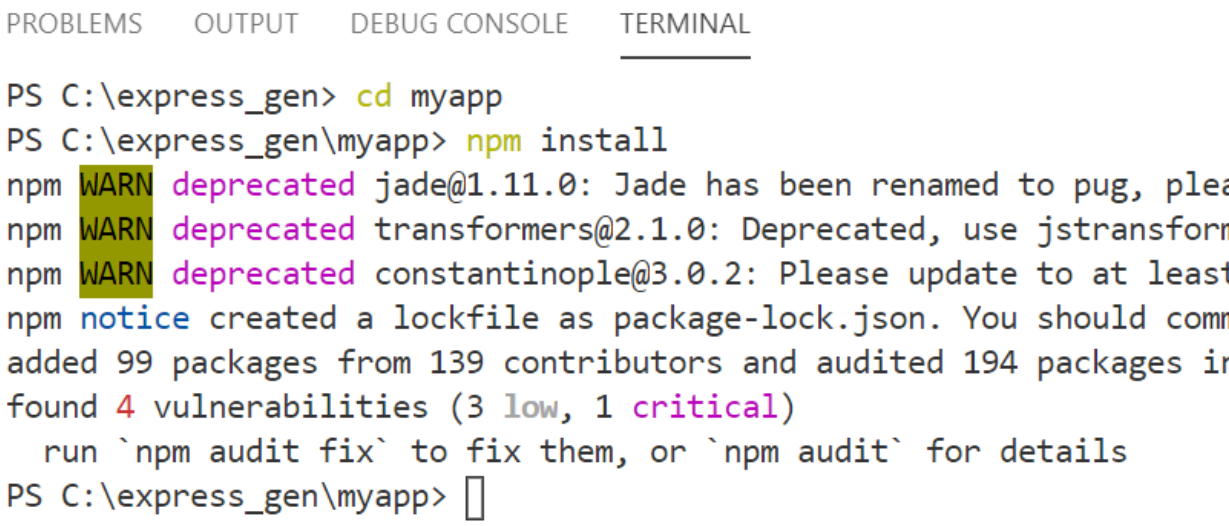
- $ express myapp



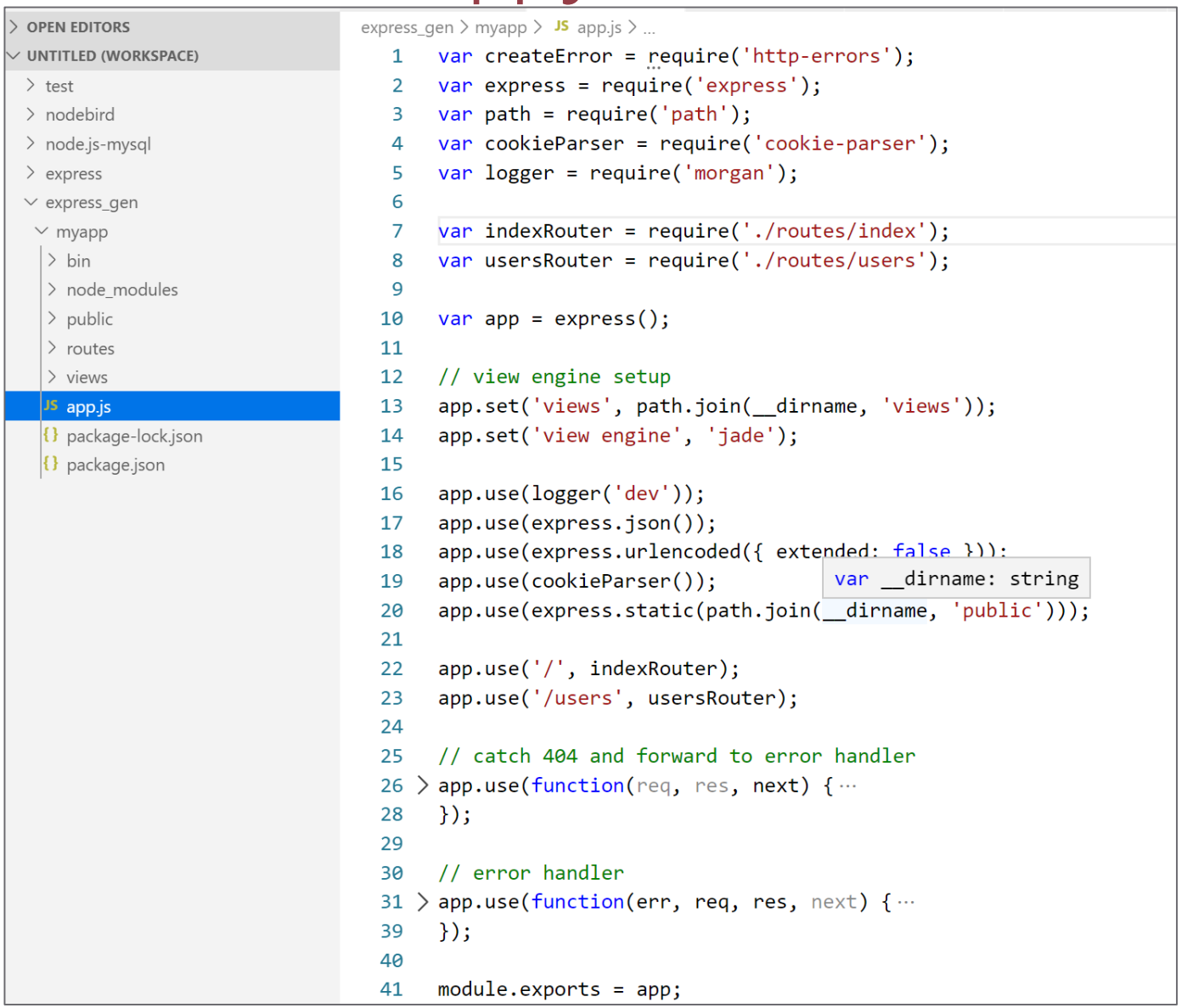
# 익스프레스 프로젝트 생성

- 프로젝트 폴더로 이동 후 필요한 모듈 설치
- $ cd myapp
- $ npm install



# 메인파일 확인 : app.js



# 실행하기

» $ npm start

```
PS C:\express_gen\myapp> npm start

> myapp@0.0.0 start C:\express_gen\myapp
> node ./bin/www

GET / 200 391.766 ms - 170
GET /stylesheets/style.css 200 7.604 ms - 111
GET / 304 14.632 ms - -
GET /stylesheets/style.css 304 1.005 ms - -
```