

JSX란?

JavaScript XML의 약자로 JavaScript를 확장한 문법이다.

- 자바스크립트에서 HTML 문법을 사용할 수 있다.
- 리액트로 프로젝트를 개발할 때 사용되므로 공식적인 자바스크립트 문법은 아니다.
- 브라우저에서 실행하기 전에 바벨을 사용하여 일반 자바스크립트 형태의 코드로 변환된다.

TSX 파일이란 무엇입니까? [↗](#)

“.tsx” 파일 확장자는 일반적으로 **React** 코드가 포함된 TypeScript 파일과 연결됩니다. **TypeScript**는 언어에 정적 타이핑을 추가하는 **JavaScript**의 상위 집합이며 React는 사용자 인터페이스 구축을 위한 JavaScript 라이브러리입니다. ******, React와 TypeScript를 함께 작업할 때 개발자는 파일에 TypeScript와 JSX(React의 JavaScript용 구문 확장)가 모두 포함되어 있음을 나타내기 위해 파일에 “.tsx” 확장자를 사용하는 경우가 많습니다.

TSX 파일 예시 [↗](#)

TypeScript를 사용하면 변수, 함수 매개변수 등에 대한 유형을 정의할 수 있습니다. React 구성 요소에 사용되는 소품, 상태 및 기타 변수의 유형을 지정하는 “.tsx” 파일에서 TypeScript 코드를 자주 볼 수 있습니다.

```
// 예: React 구성 요소의 TypeScript 코드
인터페이스 MyComponentProps {
  이름: 문자열;
  나이: 숫자;
}

const MyComponent: React.FC<MyComponentProps> = ({ 이름, 나이 }) => {
  // 여기에 컴포넌트 로직이 있습니다.

  return <div>{name}은(는) {age}세입니다.</div>;
};
```

- 먼저 react 폴더를 생성한 후 폴더 안에 들어가

\$ npx create-react-app board --template typescript

\$ cd board

\$ npm start

7. src/App.tsx 수정

```
import './App.css';

function App() {
  return (
    <div className="App">
      Hello World!
    </div>
  );
}

export default App;
```

8. 변경사항 확인

Hello World!

먼저, react-bootstrap에서 제공하는 컴포넌트를 사용하여 UI를 만들어보자.

react-bootstrap 설치

```
npm i react-bootstrap bootstrap
```

기본 bootstrap은 필수 설치 요소는 아니지만, Bootstrap Sass 파일을 사용자 정의할 계획이거나 스타일시트에 CDN을 사용하지 않으려면 기본 bootstrap도 설치하는 것이 도움이 될 수 있다.

App.tsx에 bootstrap css 추가

App.tsx 파일 상단에 아래 코드를 추가한다.

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

목록 조회 컴포넌트 생성

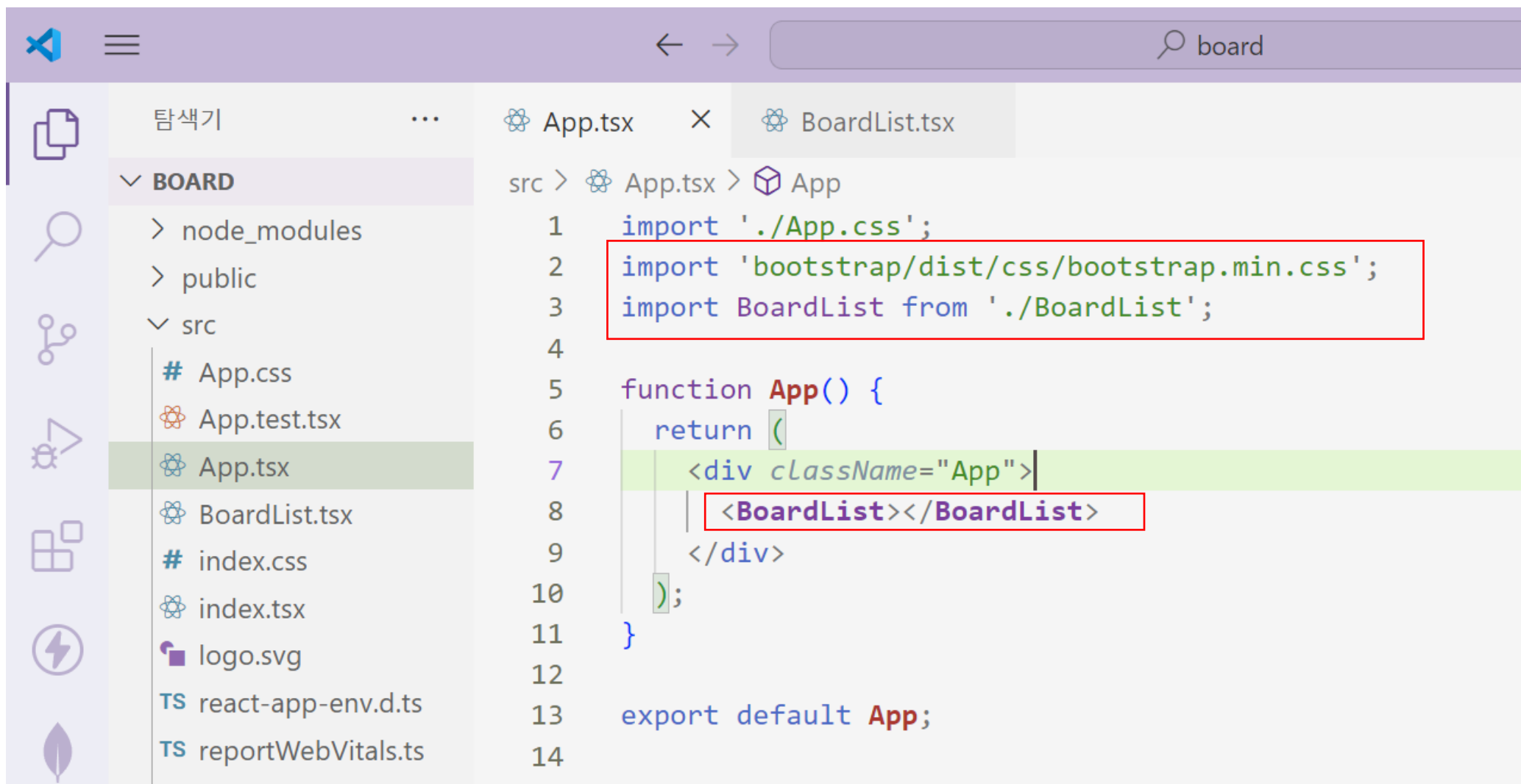
1. src 폴더 내에 BoardList.tsx 파일을 생성하고, 아래 코드를 입력한다.

```
import { Component } from "react";
import Table from "react-bootstrap/Table";

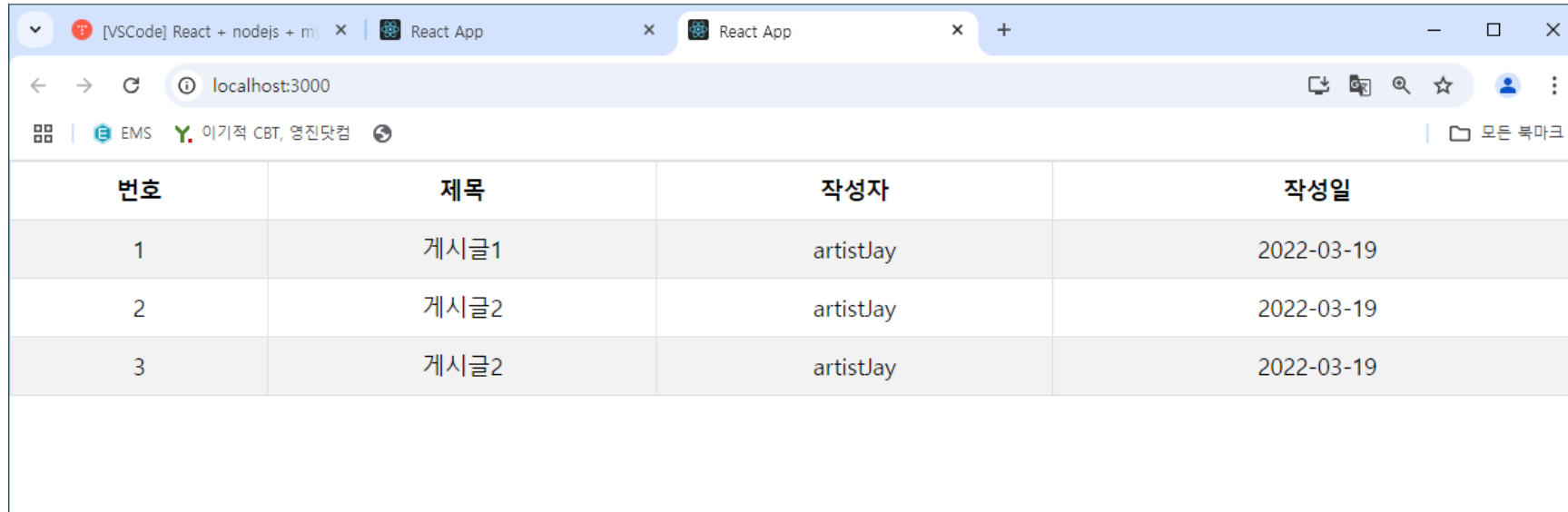
/**
 * BoardList class
 */
class BoardList extends Component {
  /**
   * @return {Component} Component
   */
  render() {
    return (
      <Table striped bordered hover>
        <thead>
          <tr>
            <th>번호</th>
            <th>제목</th>
            <th>작성자</th>
            <th>작성일</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>1</td>
            <td>게시글1</td>
            <td>artistJay</td>
            <td>2022-03-19</td>
          </tr>
          <tr>
            <td>2</td>
            <td>게시글2</td>
            <td>artistJay</td>
            <td>2022-03-19</td>
          </tr>
          <tr>
            <td>3</td>
            <td>게시글2</td>
            <td>artistJay</td>
            <td>2022-03-19</td>
          </tr>
        </tbody>
      </Table>
    );
  }
}

export default BoardList;
```

App.tsx 수정



<http://localhost:3000> 결과 화면



The screenshot shows a web browser window with two tabs. The active tab is titled 'React App' and shows a table at the URL 'localhost:3000'. The table has four columns: '번호' (Number), '제목' (Title), '작성자' (Author), and '작성일' (Date). There are three rows of data in the table.

번호	제목	작성자	작성일
1	게시글1	artistJay	2022-03-19
2	게시글2	artistJay	2022-03-19
3	게시글2	artistJay	2022-03-19

목록에 checkbox 와 '글쓰기', '수정하기', '삭제하기' 버튼을 추가하기 위해 BoardList.tsx 파일을 수정

App.tsx

BoardList.tsx ✕

src > BoardList.tsx > ...

```
1  import { Component } from "react";
2  import Table from "react-bootstrap/Table";
3  import Button from "react-bootstrap/Button";
4
5  /**
6   * BoardList class
7   */
8  class BoardList extends Component {
9    /**
10     * @return {Component} Component
11     */
12    render() {
13      ...
14      return (
15        <div>
16          <Table striped bordered hover>
17            <thead>
18              <tr>
19                <th>선택</th>
20                <th>번호</th>
21                <th>제목</th>
22                <th>작성자</th>
23                <th>작성일</th>
24
```

```
25    <tbody>
26      <tr>
27        <td>
28          <input type="checkbox"></input>
29        </td>
30        <td>1</td>
31        <td>게시글1</td>
32        <td>artistJay</td>
33        <td>2022-03-19</td>
34      </tr>
35      <tr>
36        <td>
37          <input type="checkbox"></input>
38        </td>
39        <td>2</td>
40        <td>게시글2</td>
41        <td>artistJay</td>
42        <td>2022-03-19</td>
43      </tr>
44      <tr>
45        <td>
46          <input type="checkbox"></input>
47        </td>
48        <td>3</td>
49        <td>게시글2</td>
50        <td>artistJay</td>
51        <td>2022-03-19</td>
52      </tr>
53    </tbody>
54  </Table>
```

```
55         <Button variant="info">글쓰기</Button>
56         <Button variant="secondary">수정하기</Button>
57         <Button variant="danger">삭제하기</Button>
58     </div>
59 );
60 }
61 }
62
63 export default BoardList;
64 |
```


수정된 결과 화면



The screenshot shows a web browser window with two tabs: '[VSCode] React + nodejs + m...' and 'React App'. The address bar shows 'localhost:3000'. The browser's bookmark bar contains 'EMS' and '이기적 CBT, 영진닷컴'. The main content area displays a table with five columns: '선택', '번호', '제목', '작성자', and '작성일'. The table contains three rows of data. Below the table are three buttons: '글쓰기' (blue), '수정하기' (grey), and '삭제하기' (red).

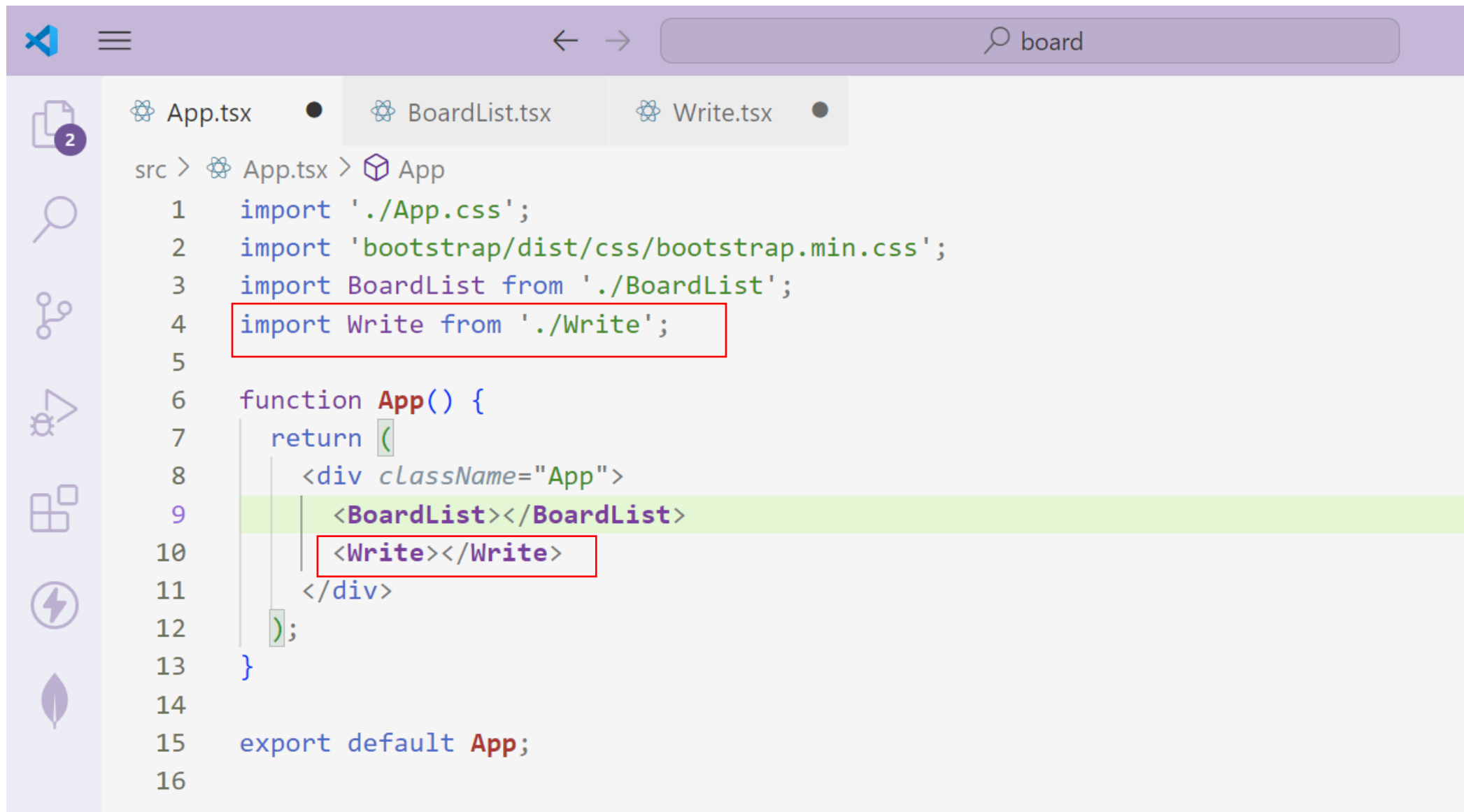
선택	번호	제목	작성자	작성일
<input type="checkbox"/>	1	게시글1	artistJay	2022-03-19
<input type="checkbox"/>	2	게시글2	artistJay	2022-03-19
<input type="checkbox"/>	3	게시글2	artistJay	2022-03-19

글쓰기 수정하기 삭제하기

글작성 컴포넌트 만들기 : src/Write.tsx 파일을 생성

```
1  import { Component } from "react";
2  import Form from "react-bootstrap/Form";
3  import { Button } from "react-bootstrap";
4  class Write extends Component {}
5  ⚡ render() {
6      return (
7          <div>
8              <Form>
9                  <Form.Group className="mb-3" controlId="exampleForm.ControlInput1">
10                     <Form.Label>제목</Form.Label>
11                     <Form.Control type="email" placeholder="name@example.com" />
12                 </Form.Group>
13                 <Form.Group className="mb-3" controlId="exampleForm.ControlTextarea1">
14                     <Form.Label>내용</Form.Label>
15                     <Form.Control as="textarea" />
16                 </Form.Group>
17             </Form>
18             <Button variant="info">작성완료</Button>
19             <Button variant="secondary">취소</Button>
20         </div>
21     );
22 }
23 }
24 export default Write;
```

App.tsx 수정



The image shows a Visual Studio Code editor window with the file `App.tsx` open. The editor has a purple header bar with the VS Code logo, a hamburger menu, and a search bar containing the text "board". Below the header, there are three tabs: `App.tsx` (active), `BoardList.tsx`, and `Write.tsx`. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, Extensions, and Testing. The main editor area shows the following code:

```
src > App.tsx > App
1  import './App.css';
2  import 'bootstrap/dist/css/bootstrap.min.css';
3  import BoardList from './BoardList';
4  import Write from './Write';
5
6  function App() {
7    return (
8      <div className="App">
9        <BoardList></BoardList>
10       <Write></Write>
11     </div>
12   );
13 }
14
15 export default App;
16
```

Red boxes highlight the import statement `import Write from './Write';` on line 4 and the `<Write></Write>` JSX element on line 10. A green highlight is also present on line 9, covering the `<BoardList></BoardList>` element.

수정된 결과 화면

[VSCode] React + nodejs + m... x React App x +

localhost:3000

EMS Y. 이기적 CBT, 영진닷컴

모든 북마크

선택	번호	제목	작성자	작성일
<input type="checkbox"/>	1	게시글1	artistJay	2022-03-19
<input type="checkbox"/>	2	게시글2	artistJay	2022-03-19
<input type="checkbox"/>	3	게시글2	artistJay	2022-03-19

글쓰기

수정하기

삭제하기

제목

name@example.com

내용

작성완료

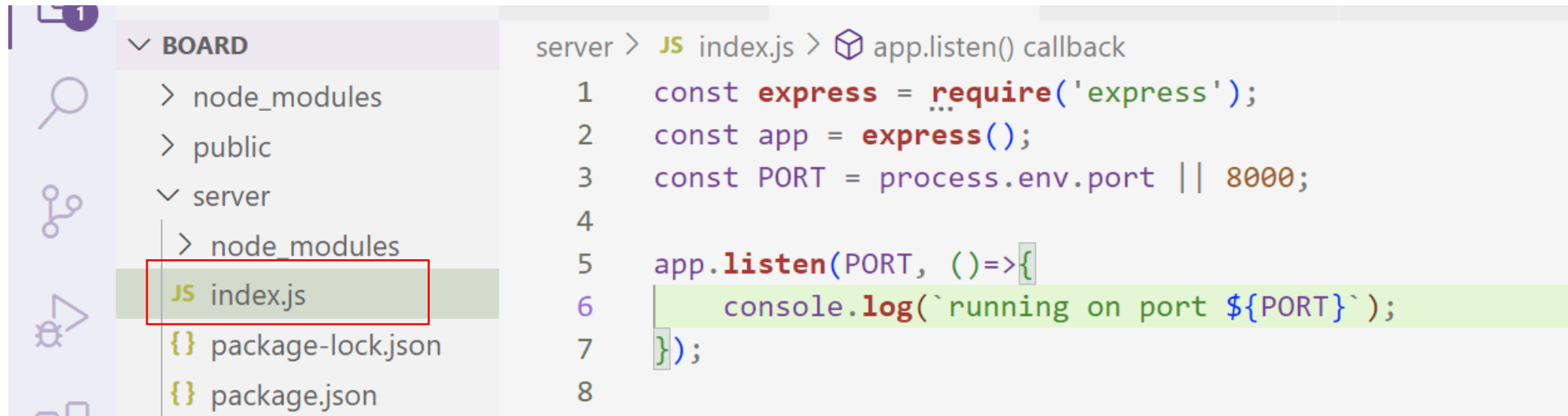
취소

서버 환경 구축 : src와 동일한 디렉토리에 server 폴더를 생성 후

```
npm init -y
```

```
npm i express body-parser mysql nodemon 설치
```

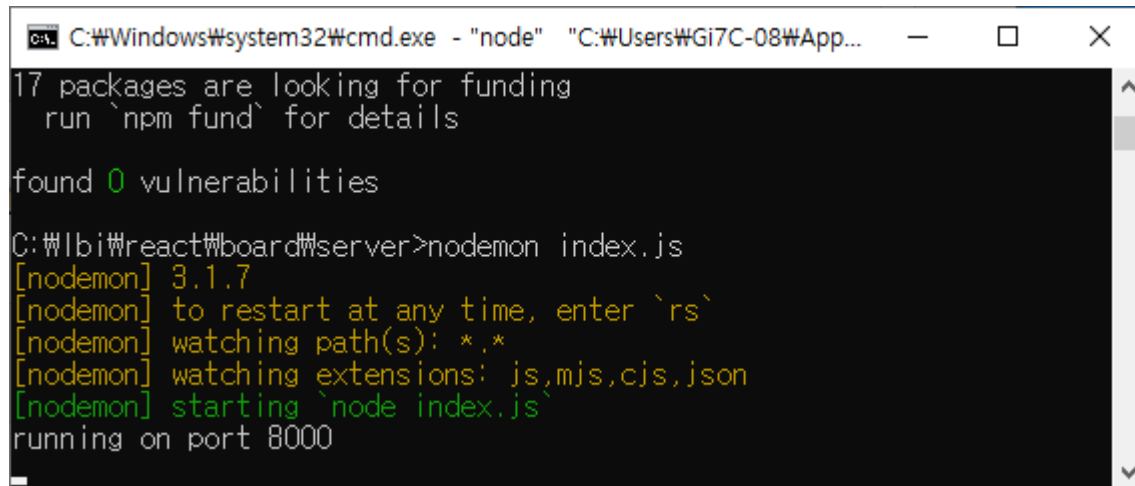
server 폴더 에 index.js 파일 생성 후 다음과 같이 작성



```
server > JS index.js > app.listen() callback
1  const express = require('express');
2  const app = express();
3  const PORT = process.env.port || 8000;
4
5  app.listen(PORT, ()=>{
6    console.log(`running on port ${PORT}`);
7  });
8
```

터미널에서 서버 실행 후

\$ nodemon index.js



```
cmd C:\Windows\system32\cmd.exe - "node" "C:\Users\Gi7C-08\AppData\Local\Microsoft\Windows\Apps\...  
17 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
  
C:\Ib\react\board\server>nodemon index.js  
[nodemon] 3.1.7  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting `node index.js`  
running on port 8000
```

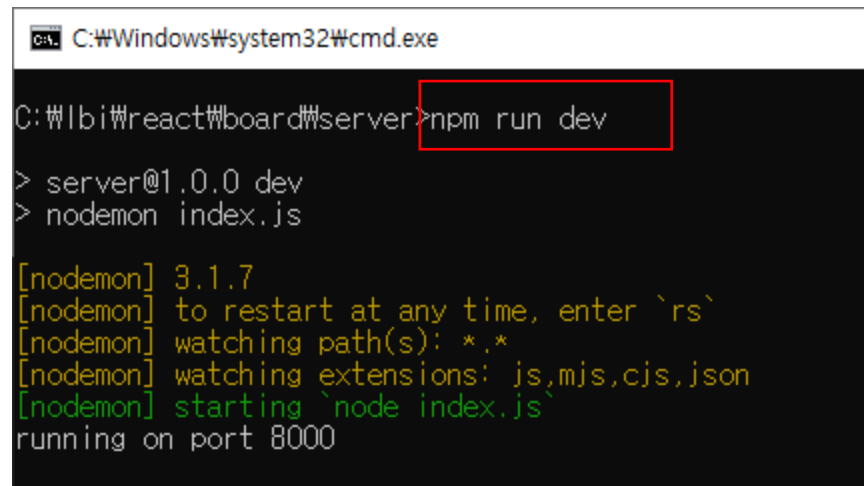
server 폴더에 있는 psckage.json 파일을 수정 후



The image shows a VS Code editor window. On the left, the file explorer shows a project structure with a 'server' folder containing 'node_modules', 'index.js', 'package-lock.json', and 'package.json'. The 'package.json' file is selected and highlighted with a red box. On the right, the code editor shows the content of 'package.json'. The 'scripts' section is highlighted with a green background, and the 'start' and 'dev' scripts are highlighted with a red box. The 'dev' script is 'nodemon index.js'.

```
server > {} package.json > {} scripts
1  {
2    "name": "server",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "start": "node index.js",
7      "dev": "nodemon index.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "description": "",
13   "dependencies": {
14     "body-parser": "^1.20.3",
15     "express": "^4.21.1",
16     "mysql": "^2.18.1",
17     "nodemon": "^3.1.7"
18   }
19 }
```

다음과 같이 실행

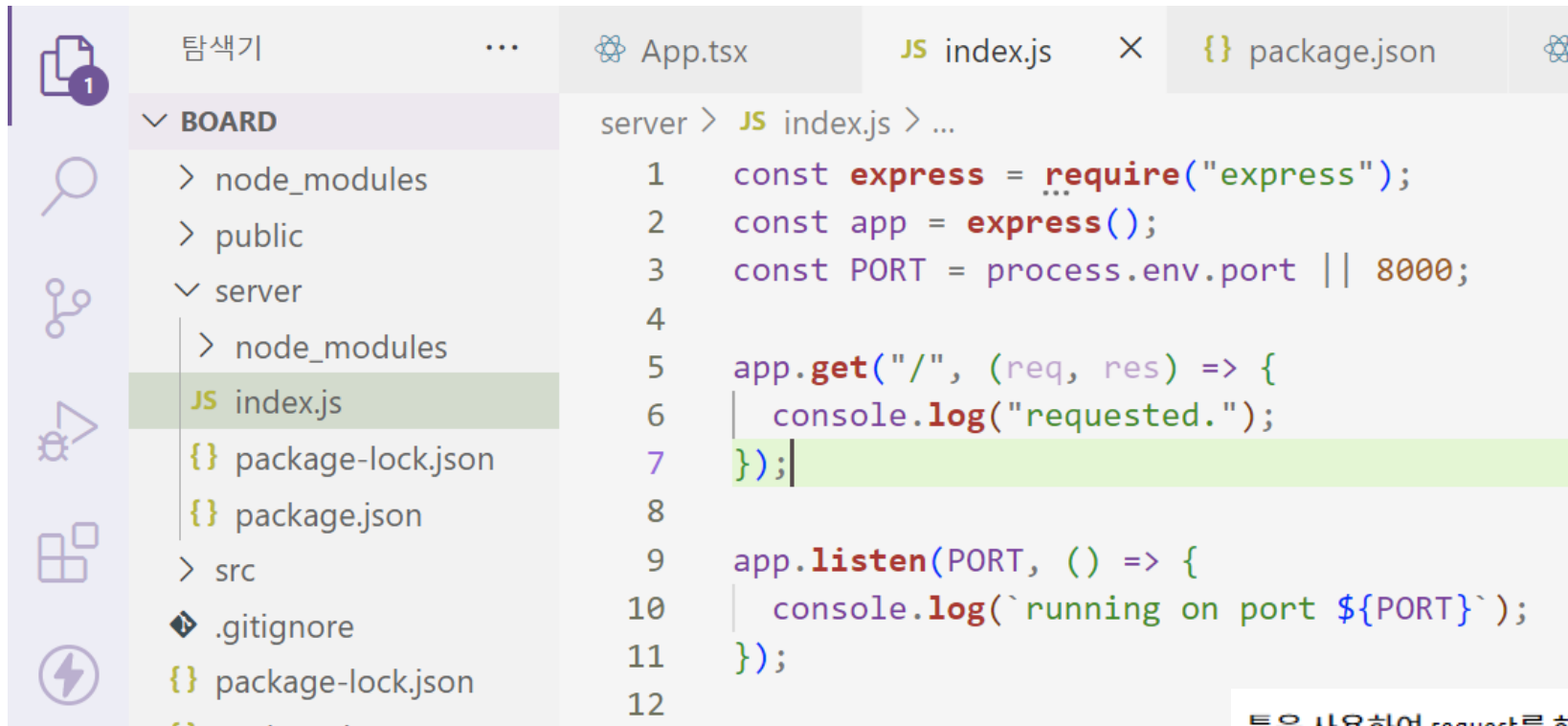


The image shows a terminal window with the command 'npm run dev' entered. The output shows the server starting on port 8000.

```
C:\Windows\system32\cmd.exe
C:\bi\react\board\server>npm run dev
> server@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
running on port 8000
```

index.js 파일 수정 후 url 테스트



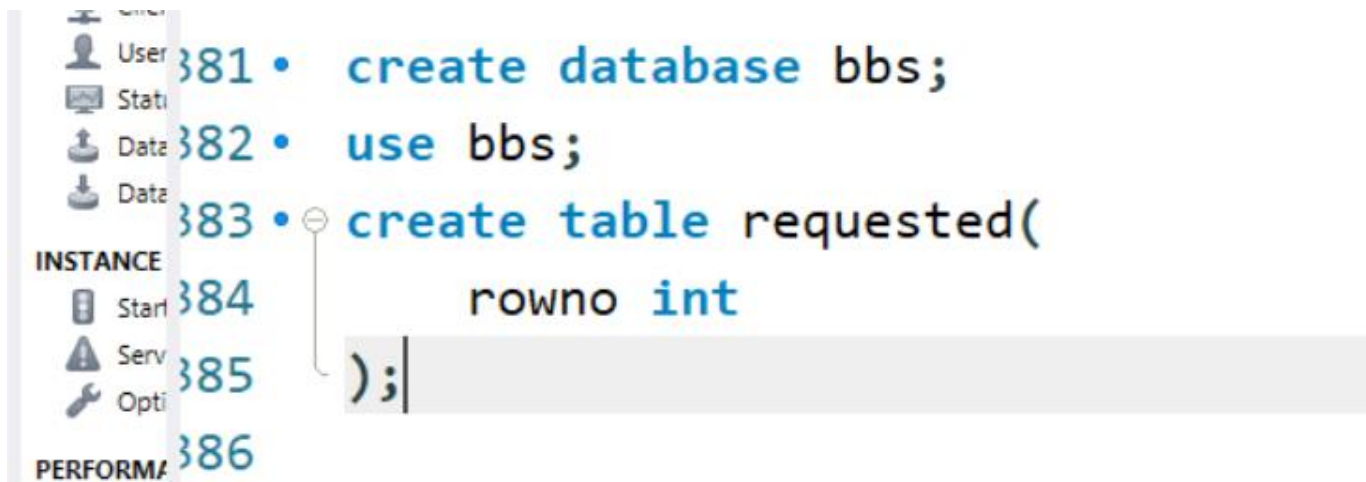
툴을 사용하여 request를 해봐도 되지만, 편하게 브라우저 상에서 URL을 호출해보자.

크롬 브라우저의 URL에 localhost:8000 입력

그리고 서버 프로젝트의 터미널에 로그가 찍히는지 확인

```
running on port 8000
requested.
requested.
requested.
requested.
requested.
requested.
```


단순히 요청이 들어오는지 카운트만 체크하기 위해 테스트용 테이블 작성



```
381 • create database bbs;
382 • use bbs;
383 • create table requested(
384     rowno int
385 );
386
```

5. 서버 프로젝트의 index.js 수정

데이터베이스 연동을 위해 index.js 파일을 수정한다.

```
const express = require("express");
const app = express();
const mysql = require("mysql");
const PORT = process.env.port || 8000;

const db = mysql.createPool({
  host: "localhost",
  user: "root",
  password: "1234",
  database: "bbs",
});

app.get("/", (req, res) => {
  const sqlQuery = "INSERT INTO requested (rowno) VALUES (1)";
  db.query(sqlQuery, (err, result) => {
    res.send("success!");
  });
});

app.listen(PORT, () => {
  console.log(`running on port ${PORT}`);
});
```

6. 정상적으로 연동이 되는지 확인

쿼리가 정상적으로 실행이 되는지 확인하기 위해 아까와 같이 브라우저에 localhost:8000을 입력해본다.

success!

테이블에 데이터가 정상적으로 들어갔는지 확인해보자.

7. Workbench에서 아래 쿼리를 실행

```
select *from requested;
```

그런데 테이블이 비어있다.

Result Grid			Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
	rowno				

만약 다음과 같은 에러가 발생한다면 아래와 같이 mysql에서 명령어 입력해 줘

실제로 에러가 나고 있었다.

ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server; consider upgrading MySQL client

구글링 해보니 해당 오류는 흔한 것 같았고, 당연히 해결방법도 있었다.

아래 명령어를 한 줄씩 순서대로 입력 후 실행하자.

```
CREATE USER 'root'@'%' IDENTIFIED BY 'root';  
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;  
FLUSH PRIVILEGES;  
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '사용할비밀번호';
```

그럼 이제 마지막으로 클라이언트에서 서버 요청 > 서버에서 데이터베이스로 데이터 insert까지 한 번에 확인을 해봐야겠지?

서버에서 데이터베이스에 insert 하는 URL은 이미 만들었으므로 클라이언트에서 그 URL을 호출만 해주면 된다.

그럼 이미 만들었던 '글쓰기' 버튼에 URL을 호출하는 로직을 테스트용으로 넣어보자.

리액트에서 서버와 통신할 때는 주로 axios 라이브러리를 사용하기 때문에 모듈을 설치해야 한다.

1. 클라이언트 프로젝트 터미널에 아래 명령어를 입력

```
npm i axios
```

2. BoardList.tsx에 axios 로직 추가

아래 코드는 BoardList class 위쪽에 추가한다.

```
import Axios from "axios";

const submitTest = () => {
  Axios.get("http://localhost:8000/", {}).then(() => {
    alert("등록 완료!");
  });
};
```

3. BoardList.tsx의 글쓰기 버튼에 아래 코드를 추가

기존

```
<Button variant="info">글쓰기</Button>
```

변경

```
<Button variant="info" onClick={submitTest}>글쓰기</Button>
```

4. 글쓰기 버튼 클릭

이번엔 CORS 오류가 발생했다.

```
✖ Access to XMLHttpRequest at 'http://localhost:8000/' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
✖ ▶ GET http://localhost:8000/ net::ERR_FAILED 200 xhr.js:210
✖ ▶ Uncaught (in promise) Error: Network Error createError.js:16
    at createError (createError.js:16:1)
    at XMLHttpRequest.handleError (xhr.js:117:1)
```

CORS 에러는 처음 서버를 세팅했을 때 아주 흔하게 발생하는 오류라서 해결방법은 별도로 포스팅했다.

<https://artistjay.tistory.com/26>

CORS란?

Cross Origin Resource Sharing의 약자로,

현재 웹페이지 도메인에서 다른 웹페이지 도메인으로 리소스가 요청되는 경우를 말한다.

예를 들면, 웹페이지인 `http://web.com`에서 API 서버 URL인 `http://api.com` 도메인으로 API를 요청하면

`http` 형태로 요청이 되므로 브라우저 자체에서 보안 상 이유로 CORS를 제한하게 되는 현상을 말합니다.

CORS 해결방법

CORS 해결방법은 클라이언트에서 처리하는 방법과 서버에서 처리하는 방법이 있다.

그런데 클라이언트에서 처리하는 경우 안 되는 케이스도 있어서 가능하면 서버에서 처리하는 방법이 안전하다.

1. 클라이언트에서 처리(withCredentials 옵션 사용)

둘 중에 한 가지 방법을 사용하면 된다.

[axios 전역 설정]

```
axios.defaults.withCredentials = true;
```

[axios 옵션 객체로 넣기]

```
axios.post(`${ServerURL}/${API}`, {},  
{  
  withCredentials: true // 쿠키 cors 통신 설정  
}).then(response => {  
  console.log(response);  
});
```

2. 서버에서 처리(Access 허용)

cors 모듈 설치가 선행되어야함

```
npm i cors
```

```
const cors = require('cors');
```

```
const express = require('express')
```

```
const cors = require('cors');
```

```
let corsOptions = {
```

```
  origin: "*", // 출처 허용 옵션
```

```
  credential: true, // 사용자 인증이 필요한 리소스(쿠키 ..등) 접근
```

```
};
```

```
app.use(cors(corsOptions));
```


게시글 번호는 수정, 삭제, 상세조회에서 사용할 고유 ID이기 때문에 빈 값이나 중복된 값을 허용해서는 안된다.

auto_increment 속성을 추가해서 자연스럽게 ID값이 증가하도록 하였다.

내용(CONTENT) 칼럼은 html을 그대로 저장하는 방법으로 진행하려고 한다.

지금은 단순 텍스트만 있으니 굳이 그럴 필요는 없겠지만 혹시 모르니까..

```
use bbs; // bbs db 사용

create table BOARD (
    BOARD_ID int not null auto_increment primary key,
    BOARD_TITLE nvarchar(30),
    BOARD_CONTENT nvarchar(500),
    REGISTER_ID nvarchar(20),
    REGISTER_DATE DATETIME DEFAULT now(),
    UPDATER_ID nvarchar(20),
    UPDATER_DATE DATETIME DEFAULT now()
);
```

그다음, 목록을 조회하는 서버 코드를 작성하자.

기존에 테스트용으로 작성했던 소스 코드와 테이블은 삭제해도 무방하다.

1. 서버 프로젝트의 index.js 파일에 아래 API 코드 추가

```
app.get("/list", (req, res) => {  
  const sqlQuery = "SELECT *FROM BOARD;";  
  db.query(sqlQuery, (err, result) => {  
    res.send(result);  
  });  
});
```

2. 게시판 테이블에 테스트 데이터 생성

Workbench에서 아래 쿼리를 한 줄씩 실행하여 조회 가능한 데이터를 생성하자.

```
insert into BOARD(BOARD_TITLE, BOARD_CONTENT, REGISTER_ID) values('제목1', '내용1', 'artistJay');  
insert into BOARD(BOARD_TITLE, BOARD_CONTENT, REGISTER_ID) values('제목2', '내용2', 'artistJay');
```

3. 생성된 데이터 확인

방금 생성한 2개의 데이터가 보이는지 확인한다.

```
select *from BOARD;
```

마지막으로 클라이언트 소스 코드를 수정하면 된다.

BoardList.tsx 파일만 수정하였다.

수정된 내용: 배열 렌더링에 사용할 Board 메서드 추가, state 추가, 목록을 가져오는 메서드 추가, componentDidMount 추가, render 함수 부분 수정

componentDidMount라는 것은 컴포넌트의 인스턴스가 생성되어 DOM 상에 삽입될 때에 호출되는 메서드 중에 하나이다. 주로, 화면이 뜬 이후에 바로 렌더링 해야 하는 데이터들(목록, 상태 조회 등)을 가져와야 할 때 해당 메서드 안에 axios를 쓰는 것 같다.

BoardList.tsx 전체 수정 파일

```
App.tsx  JS index.js  {} package.json  BoardList.tsx X
src > BoardList.tsx > Board
1  import { Component } from "react";
2  import Axios from "axios";
3  import Table from "react-bootstrap/Table";
4  import Button from "react-bootstrap/Button";
5
6  const Board = ({
7    id,
8    title,
9    registerId,
10   registerDate,
11 }): {
12   id: number;
13   title: string;
14   registerId: string;
15   registerDate: string;
16 }) => {
17   return (
18     <tr>
19       <td>
20         <input type="checkbox"/>
21       </td>
22       <td>{id}</td>
23       <td>{title}</td>
24       <td>{registerId}</td>
```

```
25         <td>{registerDate}</td>
26     </tr>
27 );
28 };
29
30 class BoardList extends Component {
31     state = {
32         boardList: [],
33     };
34
35     getList = () => {
36         Axios.get("http://localhost:8000/list", {})
37             .then((res) => {
38                 const { data } = res;
39                 this.setState({
40                     boardList: data,
41                 });
42             })
43             .catch((e) => {
44                 console.error(e);
45             });
46     };
```

```
47
48   componentDidMount() {
49     this.getList();
50   }
51
52   render() {
53     // eslint-disable-next-line
54     const { boardList }: { boardList: any } = this.state;
55
56     return (
57       <div>
58         <Table striped bordered hover>
59           <thead>
60             <tr>
61               <th>선택</th>
62               <th>번호</th>
63               <th>제목</th>
64               <th>작성자</th>
65               <th>작성일</th>
66             </tr>
67           </thead>
68           <tbody>
```

```

69     {
70         // eslint-disable-next-line
71         boardList.map((v: any) => {
72             return (
73                 <Board
74                     id={v.BOARD_ID}
75                     title={v.BOARD_TITLE}
76                     registerId={v.REGISTER_ID}
77                     registerDate={v.REGISTER_DATE}
78                 />
79             );
80         })
81     }
82 </tbody>
83 </Table>
84 <Button variant="info">글쓰기</Button>
85 <Button variant="secondary">수정하기</Button>
86 <Button variant="danger">삭제하기</Button>
87 </div>
88 );
89 }
90 }
91 export default BoardList;

```

코드 수정 후 클라이언트와 서버 모두 재 실행 해주어야 함

작성일을 좀더 간결하게 표현하기 위해 서버 프로젝트의 index.js 파일을 수정

```
// 기존
app.get("/list", (req, res) => {
  const sqlQuery = "SELECT *FROM BOARD;";
  db.query(sqlQuery, (err, result) => {
    res.send(result);
  });
});

// 변경
app.get("/list", (req, res) => {
  const sqlQuery = "SELECT BOARD_ID, BOARD_TITLE, REGISTER_ID, DATE_FORMAT(REGISTER_DATE, '%Y-%m-%d') AS REGISTER_DATE FROM BOARD;";
  db.query(sqlQuery, (err, result) => {
    res.send(result);
  });
});
```

그리고 화면을 refresh 해서 변경사항을 확인한다.

선택	번호	제목	작성자	작성일
<input type="checkbox"/>	1	제목1	artistJay	2022-03-22
<input type="checkbox"/>	2	제목2	artistJay	2022-03-22

글쓰기

수정하기

삭제하기

제목

name@example.com

내용

작성완료

취소

끝인 줄 알았는데 콘솔을 확인해보니, 에러가 발생하고 있었다.

Each child in a list should have a unique "key" prop.

원인

React는 key prop을 사용하여 컴포넌트와 DOM 요소 간의 관계를 생성한다. 리액트 라이브러리는 이 관계를 이용해 컴포넌트 리 렌더링 여부를 결정한다. 불필요한 리 렌더링을 방지하기 위해서는 각 자식 컴포넌트마다 독립적인 key값을 넣어줘야 한다.

해결방법

간단하게 렌더링 하는 부분에 key 값을 넣어주었다.

```
boardList.map((v: any) => {  
  return (  
    <Board  
      id={v.BOARD_ID}  
      title={v.BOARD_TITLE}  
      registerId={v.REGISTER_ID}  
      registerDate={v.REGISTER_DATE}  
      key={v.BOARD_ID}  
    />  
  );  
})
```

'글쓰기' 버튼과 '취소' 버튼은 현재 시점에서는 필요가 없다. 나중에 화면 단위로 구성할 때 구현하겠다.

- a) 작성 완료 버튼 event 추가
- b) 제목과 내용을 DB에 insert/update 하는 서버 API 코드 작성
- c) 수정하기 버튼 event 추가
- d) 게시글 상세 조회 select 하는 서버 API 코드 작성
- e) 삭제하기 버튼 event 추가
- f) 게시글 삭제 delete 하는 서버 API 코드 작성

1. 작성 완료 버튼 event 추가

작성 완료 버튼은 2가지의 기능을 동시에 가지고 있다. 수정인지 새로운 글인지에 대한 플래그 값을 두어서 분기 처리할 예정이다.

Write.tsx 파일에 아래 소스 코드를 추가한다.

Write.tsx 수정

```
1  import { Component } from "react";
2  import Axios from "axios";
3  import Form from "react-bootstrap/Form";
4  import Button from "react-bootstrap/Button";
5
6  interface IProps {
7    isModifyMode: boolean;
8    boardId: number;
9    handleCancel: any;
10 }
11
12 class Write extends Component<IProps> {
13   constructor(props: any) {
14     super(props);
15     this.state = {
16       title: "",
17       content: "",
18       isRendered: false,
19     };
20   }
21
22   state = {
23     title: "",
24     content: "",
25     isRendered: false,
26   };
27 }
```



```
28     write = () => {
29         Axios.post("http://localhost:8000/insert", {
30             title: this.state.title,
31             content: this.state.content,
32         })
33         .then((res) => {
34             this.setState({
35                 title: "",
36                 content: "",
37             });
38             this.props.handleCancel();
39         })
40         .catch((e) => {
41             console.error(e);
42         });
43     };
44
```



```
44
45     update = () => {
46         Axios.post("http://localhost:8000/update", {
47             title: this.state.title,
48             content: this.state.content,
49             id: this.props.boardId,
50         })
51         .then((res) => {
52             this.setState({
53                 title: "",
54                 content: "",
55             });
56             this.props.handleCancel();
57         })
58         .catch((e) => {
59             console.error(e);
60         });
61     };
62
```



```
63   detail = () => {
64     Axios.get(`http://localhost:8000/detail?id=${this.props.boardId}`)
65       .then((res) => {
66         if (res.data.length > 0) {
67           this.setState({
68             title: res.data[0].BOARD_TITLE,
69             content: res.data[0].BOARD_CONTENT,
70           });
71         }
72       })
73       .catch((e) => {
74         console.error(e);
75       });
76   };
77
78   handleChange = (e: any) => {
79     this.setState({
80       [e.target.name]: e.target.value,
81     });
82   };
83
```




```
84   componentDidUpdate = (prevProps: any) => {
85     if (this.props.isModifyMode && this.props.boardId !== prevProps.boardId) {
86       this.detail();
87     }
88   };
89
90   render() {
91     return (
92       <div>
93         <Form>
94           <Form.Group className="mb-3">
95             <Form.Label>제목</Form.Label>
96             <Form.Control
97               type="text"
98               name="title"
99               value={this.state.title}
100               onChange={this.handleChange}
101               placeholder="제목을 입력하세요"
102             />
103           </Form.Group>
104           <Form.Group className="mb-3">
105             <Form.Label>내용</Form.Label>
106             <Form.Control
107               as="textarea"
108               name="content"
109               value={this.state.content}
110               onChange={this.handleChange}
111               placeholder="내용을 입력하세요"
112             />
113           </Form.Group>
```



```
114         </Form>
115         <Button variant="info" onClick={this.props.isModifyMode ? this.update : this.write}>
116             |   작성완료
117         </Button>
118         <Button variant="secondary" onClick={this.props.handleCancel}>
119             |   취소
120         </Button>
121     </div>
122     );
123 }
124 }
125
126 export default Write;
```

서버 폴더의 index.js 수정



The image shows a code editor interface. On the left is a file explorer with a tree view. The 'BOARD' folder is expanded, showing subfolders 'node_modules', 'public', and 'server'. The 'server' folder is further expanded, showing 'node_modules', 'index.js' (highlighted), 'package-lock.json', 'package.json', and a 'src' folder. The 'src' folder contains various files like 'App.css', 'App.test.tsx', 'App.tsx', 'BoardList.tsx', 'index.css', 'index.tsx', 'logo.svg', and several TypeScript files.

The main editor area shows the content of 'index.js'. The code is as follows:

```
server > JS index.js > ...
1  const express = require("express");
2  const app = express();
3  const mysql = require("mysql");
4  const PORT = process.env.port || 8000;
5
6  const cors = require('cors');
7
8  const bodyParser = require("body-parser");
9
10
11  let corsOptions = {
12    origin: "*", // 출처 허용 옵션
13    credential: true, // 사용자 인증이 필요한 리소스(쿠키 .. 등) 접근
14  };
15
16  app.use(express.json());
17  app.use(bodyParser.urlencoded({ extended: true }));
18
19  app.use(cors(corsOptions));
20
```

Two red boxes highlight specific lines of code: the first box highlights line 8, `const bodyParser = require("body-parser");`, and the second box highlights lines 16 and 17, `app.use(express.json());` and `app.use(bodyParser.urlencoded({ extended: true }));`.

```
44
45 app.post("/insert", (req, res) => {
46   var title = req.body.title;
47   var content = req.body.content;
48
49   const sqlQuery =
50   | "insert into BOARD(BOARD_TITLE, BOARD_CONTENT, REGISTER_ID) values(?, ?, 'artistJay');"
51   db.query(sqlQuery, [title, content], (err, result) => {
52     res.send(result);
53   });
54 });
55
56 app.post("/update", (req, res) => {
57   var title = req.body.title;
58   var content = req.body.content;
59
60   const sqlQuery =
61   | "UPDATE BOARD SET BOARD_TITLE = ?, BOARD_CONTENT = ?, UPDATER_ID) FROM (?, ?, artistJay);"
62   db.query(sqlQuery, [title, content], (err, result) => {
63     res.send(result);
64   });
65 });
66
```

그럼 BoardList.tsx 파일만 수정해보자.

1. 삭제하기 버튼에 클릭 event를 추가한다.

```
<Button variant="danger" onClick={this.handleDelete}>  
  삭제하기  
</Button>
```

그리고 handleDelete 메서드도 만들어준다.

```
handleDelete = () => {  
  if (this.state.checkList.length == 0) {  
    alert("삭제할 게시글을 선택하세요.");  
    return;  
  }  
  
  let boardIdList = "";  
  
  this.state.checkList.forEach((v: any) => {  
    boardIdList += `${v}`,`  
  });  
  
  Axios.post("http://localhost:8000/delete", {  
    boardIdList: boardIdList.substring(0, boardIdList.length - 1),  
  })  
    .then(() => {  
      this.getList();  
    })  
    .catch((e) => {  
      console.error(e);  
    });  
};
```

2. 게시물 삭제 서버 API 코드를 작성한다.

index.js

```
67  app.post("/delete", (req, res) => {
68    const id = req.body.boardIdList; // 6,5
69
70    const sqlQuery = `DELETE FROM BOARD WHERE BOARD_ID IN (${id})`;
71    db.query(sqlQuery, (err, result) => {
72      res.send(result);
73    });
74  });
```