axios로 todo앱 구현 #2

**학습목표**

json-server + axios + useState 조합으로 todo앱 구현하기

**목차**

[1. todo6 2](#_Toc150943281)

[1) 개요 2](#_Toc150943282)

[2) 프로젝트 생성 2](#_Toc150943283)

[3) 구현 3](#_Toc150943284)

# todo6

## 개요

useState + axios 조합으로 todo 앱을 구현하자.

## 프로젝트 생성

|  |
| --- |
| npx create-react-app todo6 --template typescript |

### axios 설치

|  |
| --- |
| npm install axios |

## 구현

### types.ts

src/types.ts

|  |
| --- |
| export type Todo = {    id: number,    title: string,    done: boolean  };  export type AddTodoFunc = (text: string) => void;  export type ToogleTodoFunc = (todo: Todo) => void;  export type DeleteTodoFunc = (id: number) => void; |

AddTodoFunc

addTodo 함수의 타입 선언

ToogleTodoFunc

toogleTodo 함수의 타입 선언

DeleteTodoFunc

deleteTodo 함수의 타입 선언

### TodoItem.tsx

src/TodoItem.tsx

|  |
| --- |
| import React from 'react'  import type \* as types from './types'  type Props = {    todo: types.Todo,    toggleTodo: types.ToogleTodoFunc,    deleteTodo: types.DeleteTodoFunc  }  function TodoItem({ todo, toggleTodo, deleteTodo }: Props) {    const onChange = () => toggleTodo(todo)    const onClick = () => {      if (window.confirm('삭제하시겠습니까?'))        deleteTodo(todo.id)    }    return (      <tr className={ todo.done ? "done" : "" }>        <td>{ todo.id }</td>        <td>          <input type='checkbox' checked={ todo.done } onChange={onChange} />          { todo.title }          <span onClick={onClick}>x</span>        </td>      </tr>    )  }  export default TodoItem; |

### TodoList.tsx

src/TodoList.tsx

|  |
| --- |
| import React from 'react'  import type \* as types from './types'  import TodoItem from './TodoItem'  type Props = {    todoList: types.Todo[],    toggleTodo: types.ToogleTodoFunc,    deleteTodo: types.DeleteTodoFunc  }  function TodoList({ todoList, toggleTodo, deleteTodo }: Props) {    const trlist = todoList.map(todo =>      <TodoItem todo={todo} key={todo.id} toggleTodo={toggleTodo} deleteTodo={deleteTodo} />);    return (      <table>        <thead>          <tr><td>id</td><td>할일</td></tr>        </thead>        <tbody>          {trlist}        </tbody>      </table>    )  }  export default TodoList |

### TodoInput.tsx

src/TodoInput.tsx

|  |
| --- |
| import React, { ChangeEvent, useState } from 'react'  import type \* as types from './types'  type Props = {    addTodo : types.AddTodoFunc  }  function TodoInput({ addTodo }: Props) {    const [title, setTitle] = useState<string>('');    const onChange = (e: ChangeEvent<HTMLInputElement>) => setTitle(e.target.value);    const onClick = () => { addTodo(title); setTitle(''); }    return (      <div>        <input type='text' value={title} onChange={onChange} />        <button onClick={onClick}>추가</button>      </div>    )  }  export default TodoInput |

### TodoRoot.tsx

src/TodoRoot.tsx todo앱 화면을 담당하는 컴포넌트

|  |
| --- |
| import React from 'react'  import './TodoRoot.css'  import type \* as types from './types'  import TodoInput from './TodoInput';  import TodoList from './TodoList';  type Props = {    todoList: types.Todo[],    addTodo: types.AddTodoFunc,    toggleTodo: types.ToogleTodoFunc,    deleteTodo: types.DeleteTodoFunc  }  function TodoRoot({ todoList, addTodo, toggleTodo, deleteTodo }: Props) {    return (      <div className="todoRoot">        <h1>할일</h1>        <TodoInput addTodo={addTodo} />        <TodoList todoList={todoList} toggleTodo={toggleTodo} deleteTodo={deleteTodo} />      </div>    )  }  export default TodoRoot; |

### TodoRoot.css

src/TodoRoot.css

|  |
| --- |
| .todoRoot { border: 1px solid gray; width: 400px; margin: 10px auto;    box-shadow: 5px 5px 5px #DDD; padding: 30px; }  .todoRoot h1 { margin: 0  0 20px 0; }  .todoRoot div > input { padding: 6px; width: 280px; }  .todoRoot div button { margin-left: 5px; padding: 3px 10px; }  .todoRoot table { border-collapse: collapse; margin-top: 20px; width: 100%; }  .todoRoot td { border: 1px solid gray; padding: 6px; }  .todoRoot thead { background-color: #eee; text-align: center; }  .todoRoot td:nth-child(1) { text-align: center; width: 30px; }  .todoRoot input[type=checkbox] { margin-right: 10px; accent-color: gray; }  .todoRoot span { float: right; font-weight: bold; cursor: pointer; }  .todoRoot tr.done { color: #bbb; text-decoration: line-through; } |

### TodoContainer.tsx

src/TodoContainer.tsx

|  |
| --- |
| import React, { useState, useEffect } from 'react'  import axios from 'axios';  import type { Todo } from './types'  import TodoRoot from './TodoRoot';  axios.defaults.baseURL = "http://localhost:3000";  function TodoContainer() {    const [todoList, setTodoList] = useState<Todo[]>([]);  async function reloadTodos() {      try {        const response = await axios.get("/todos");        const todos: Todo[] = response.data;        setTodoList(todos);      } catch (error) {        if (error instanceof Error) alert('조회 에러: ' + error.message);        else alert('조회 에러: ' + error);      }    }    useEffect(() => {      reloadTodos();    }, []);    async function addTodo(title: string) {      try {        const todo = {title: title, done: false};        await axios.post("/todos/", todo);        reloadTodos();      } catch (error) {        if (error instanceof Error) alert('저장 에러: ' + error.message);        else alert('저장 에러: ' + error);      }    }    async function toogleTodo(todo: Todo) {      try {        await axios.put("/todos/" + todo.id, { ...todo, done: !todo.done });        reloadTodos();      } catch (error) {        if (error instanceof Error) alert('저장 에러: ' + error.message);        else alert('저장 에러: ' + error);      }    }    async function deleteTodo(id: number) {      try {        await axios.delete("/todos/" + id);        reloadTodos();      } catch (error) {        if (error instanceof Error) alert('삭제 에러: ' + error.message);        else alert('삭제 에러: ' + error);      }    }    return <TodoRoot todoList={todoList}                     addTodo={addTodo}                     toggleTodo={toogleTodo}                     deleteTodo={deleteTodo} />;  }  export default TodoContainer |

### App.tsx

|  |
| --- |
| import React from 'react';  import TodoRoot from './TodoRoot';  function App() {    return (      <TodoRoot />    );  }  export default App; |

### 실행

터미널 창 #1 - 백엔드 실행

|  |
| --- |
| json-server --watch todo-db.json |

위 명령은 todo-db.json 파일이 있는 src 디렉토리에서 실행해야 한다.

터미널 창 #2 - 프론트엔드 플젝 실행

|  |
| --- |
| npm run start |