React

◆ 일정관리 앱

정수아

Contents

01 프로젝트 준비하기

프로젝트 준비하기

❖ 프로젝트 생성

```
$ npx create-react-app todo-app
```

❖ 라이브러리 설치

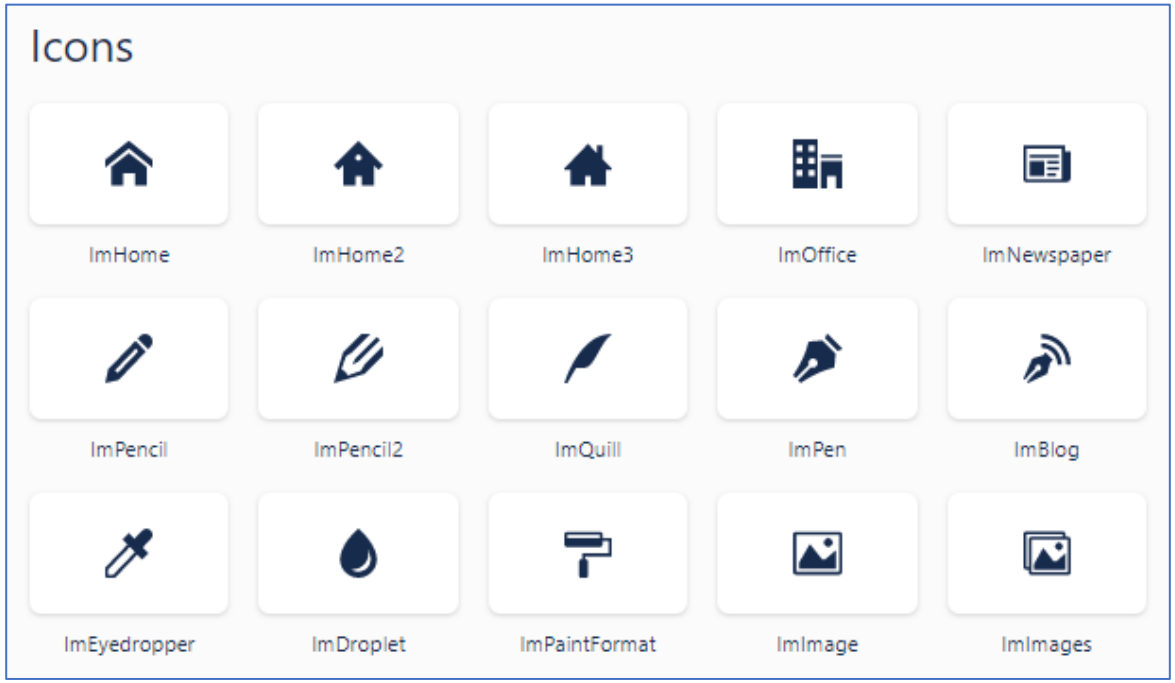
```
$ npm install --save react-icons
```

- react-icons
  - https://react-icons.github.io/react-icons

프로젝트 준비하기

❖ **react-icons**

- https://react-icons.netlify.com
- 리액트에서 사용할 수 있는 아이콘 라이브러리
- 아이콘을 리액트 컴포넌트처럼 사용할 수 있음



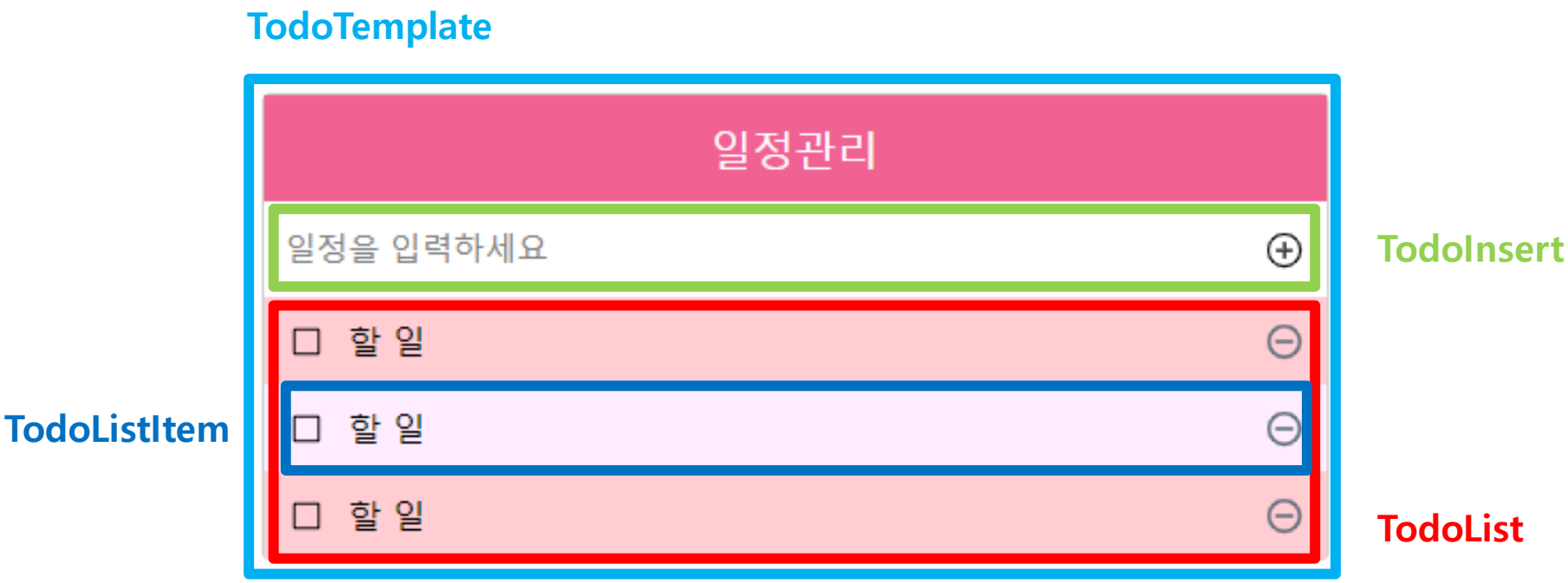
프로젝트 준비하기

❖ **일정관리 어플리케이션에서 필요한 기능 정의**

- 아이템 추가
- 아이템 삭제
- 전체 아이템 목록
- 아이템 체크박스
- 아이템 필터링

프로젝트 준비하기

❖ 프로젝트 구조



프로젝트 준비하기

❖ 프로젝트 구조

프로젝트 준비하기

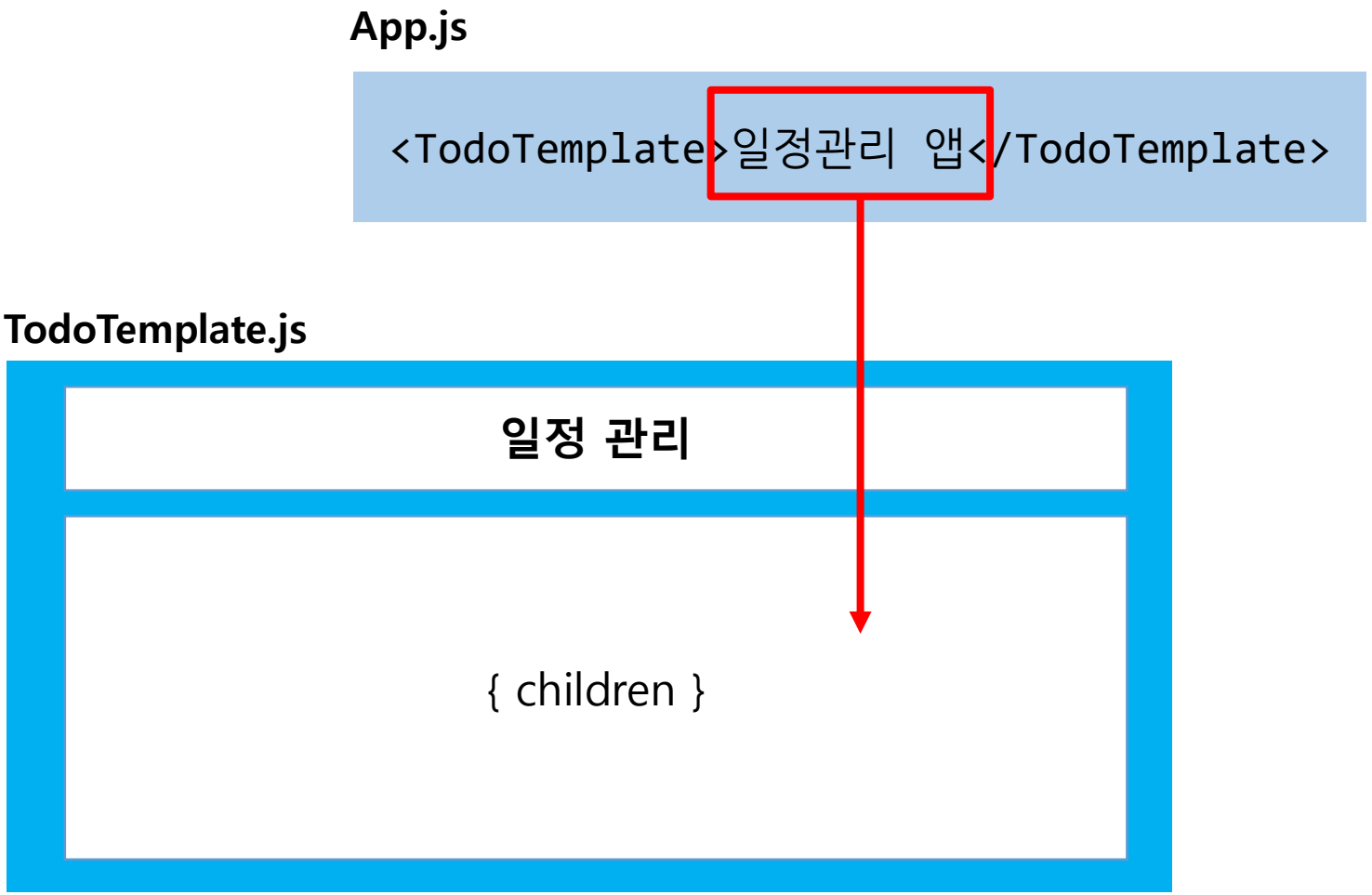
❖ **필요한 파일 준비**

- components 폴더
  - TodoTemplate.js
  - TodoInsert.js
  - TodoList.js
  - TodoListItem.js

- style 폴더
  - TodoTemplate.scss
  - TodoInsert.scss
  - TodoList.scss
  - TodoListItem.scss

02 UI 구성하기

1. 전체적인 틀 만들기


App.js
<TodoTemplate>일정관리 앱</TodoTemplate>
TodoTemplate.js
일정 관리
{ children }

1. 전체적인 틀 만들기

❖ App.js

```
import './App.css';
import TodoTemplate from './components/TodoTemplate';

function App() {
  return (
    <TodoTemplate>일정관리 앱</TodoTemplate>
  );
}

export default App;
```

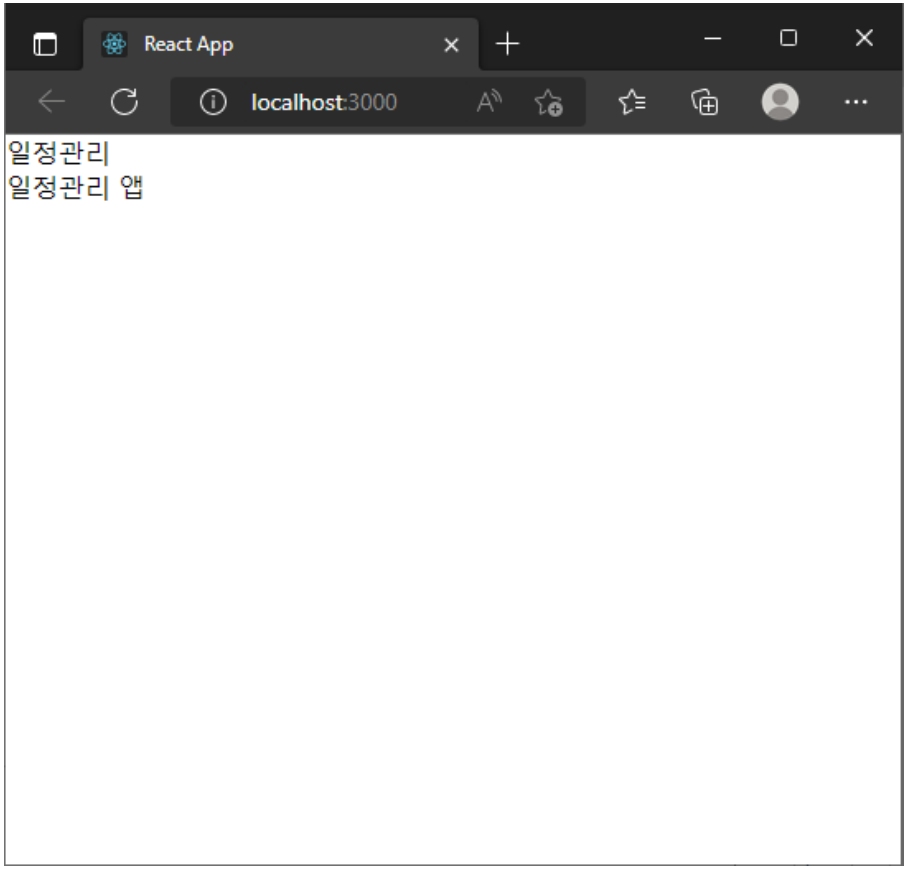
1. 전체적인 틀 만들기

❖ TodoTemplate.js

```
import React from 'react';
import '../style/TodoTemplate.scss';

const TodoTemplate = ({children}) => {
    return (
        <div className="TodoTemplate">
            <div className="appTitle">일정관리</div>
            <div className="content">{children}</div>
        </div>
    );
};

export default TodoTemplate;
```

1. 전체적인 틀 만들기

❖ 실행 결과



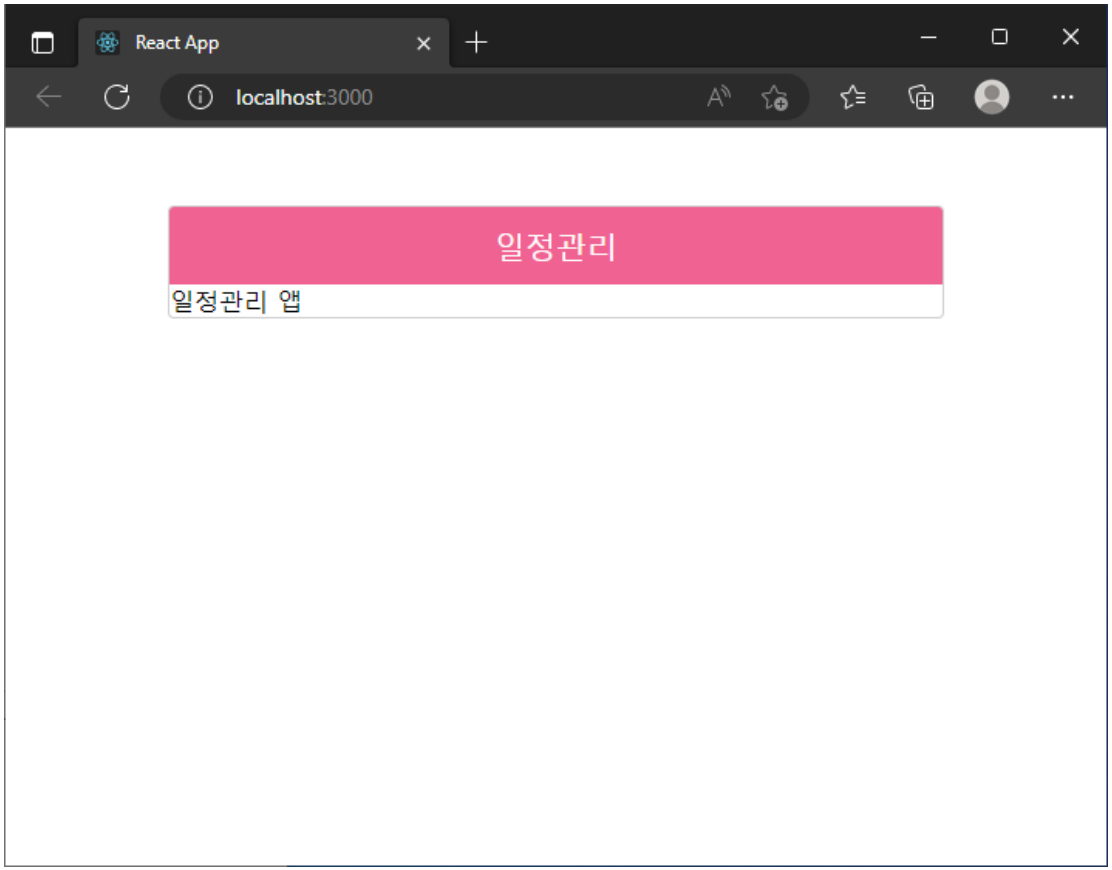
1. 전체적인 틀 만들기

❖ TodoTemplate.scss

```
.TodoTemplate {
    border: 1px solid #cfcfcf;
    width : 500px;
    margin : 0 auto;
    margin-top: 50px;
    border-radius: 4px;
    overflow: hidden;

    .appTitle {
        background: #f06292;
        color: white;
        height: 50px;
        font-size: 20px;
        display: flex;
        justify-content: center;
        align-items: center;
    }
}
```

1. 전체적인 틀 만들기

❖ 실행 결과

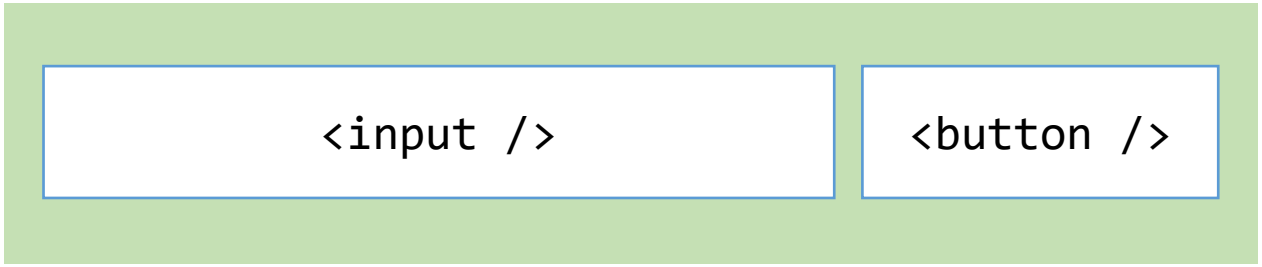


2. 입력 및 추가 부분 만들기

App.js

```
<TodoTemplate>
    <TodoInsert/>
</TodoTemplate>
```

TodoInsert.js



2. 입력 및 추가 부분 만들기

❖ App.js

```
import './App.css';
import TodoTemplate from './components/TodoTemplate';
import TodoInsert from './components/TodoInsert';

function App() {
  return (
    <TodoTemplate>
        <TodoInsert/>
    </TodoTemplate>
  );
}

export default App;
```

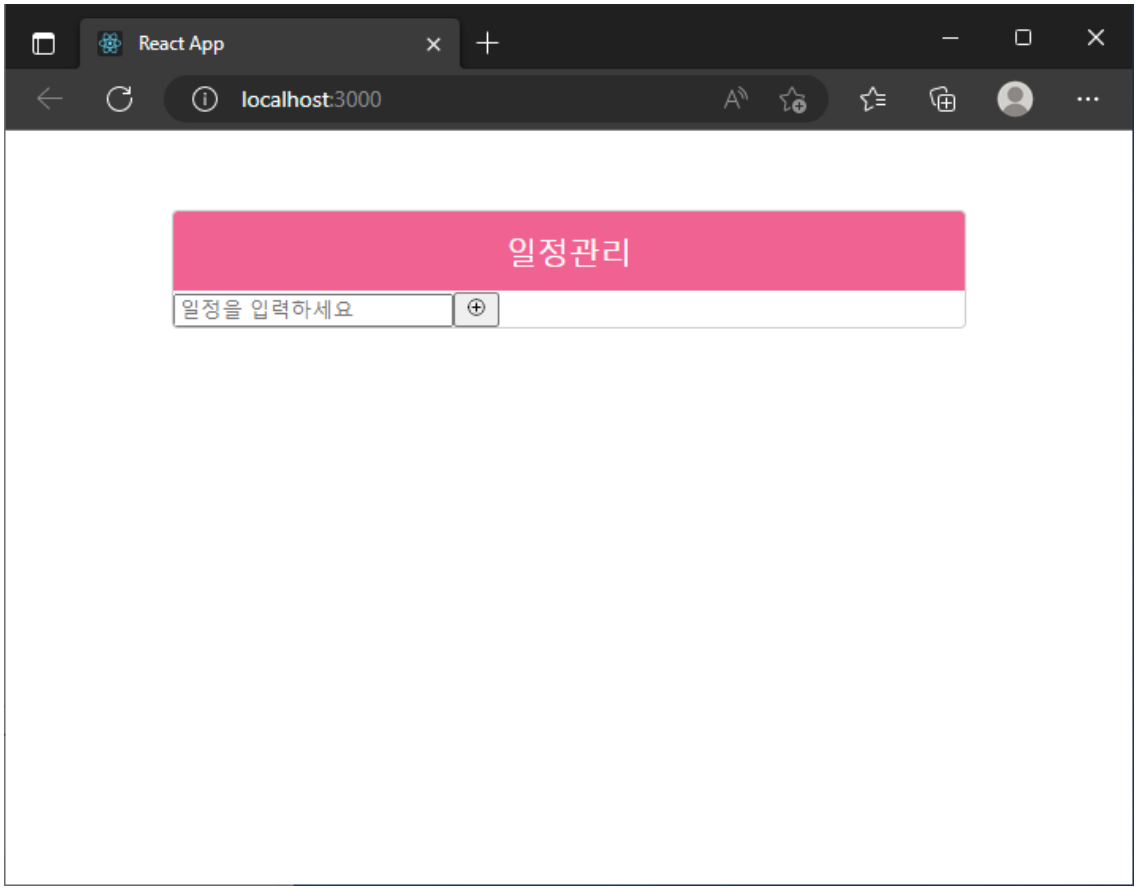
2. 입력 및 추가 부분 만들기

❖ TodoInsert.js

```
import React from 'react';
import '../style/TodoInsert.scss';
import {IoIosAddCircleOutline} from 'react-icons/io';

const TodoInsert = () => {
    return (
        <form className="TodoInsert">
            <input type="text" placeholder="일정을 입력하세요"/>
            <button type="submit">
                <IoIosAddCircleOutline/>
            </button>
        </form>
    );
};

export default TodoInsert;
```

2. 입력 및 추가 부분 만들기

❖ 실행 결과



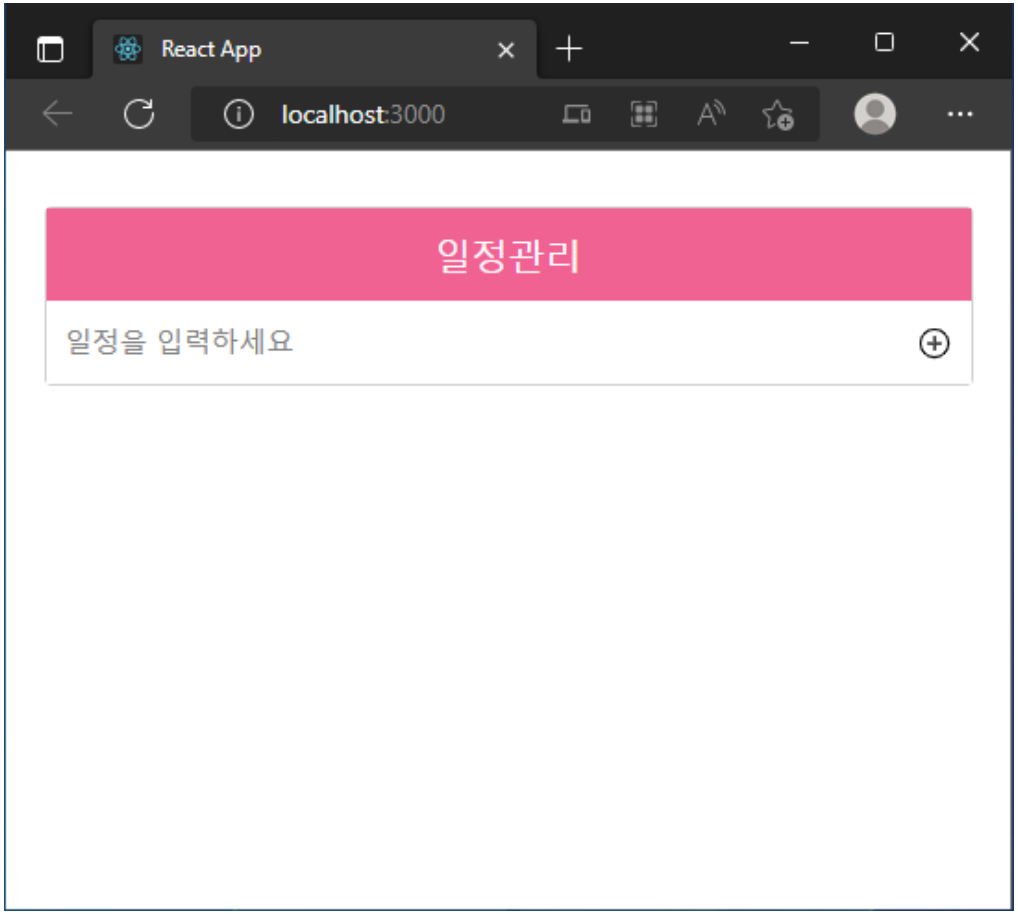
2. 입력 및 추가 부분 만들기

❖ TodoInsert.scss

```scss
.TodoInsert {
    display: flex;
    background : white;

    input {
        background: none;
        outline: none;
        border: none;
        padding: 10px;
        font-size: 15px;
        line-height: 25px;
        color: black;
        flex: 1;
        &::placeholder {
            color: grey;
        }
    }
    button {
        background: none;
        outline: none;
        border: none;
        color: black;
        padding: 10px;
        font-size: 20px;
        display: flex;
        align-items: center;
        cursor: pointer;
        &:hover {
            background: #f7bacf;
        }
    }
}
```

2. 입력 및 추가 부분 만들기

❖ 실행 결과

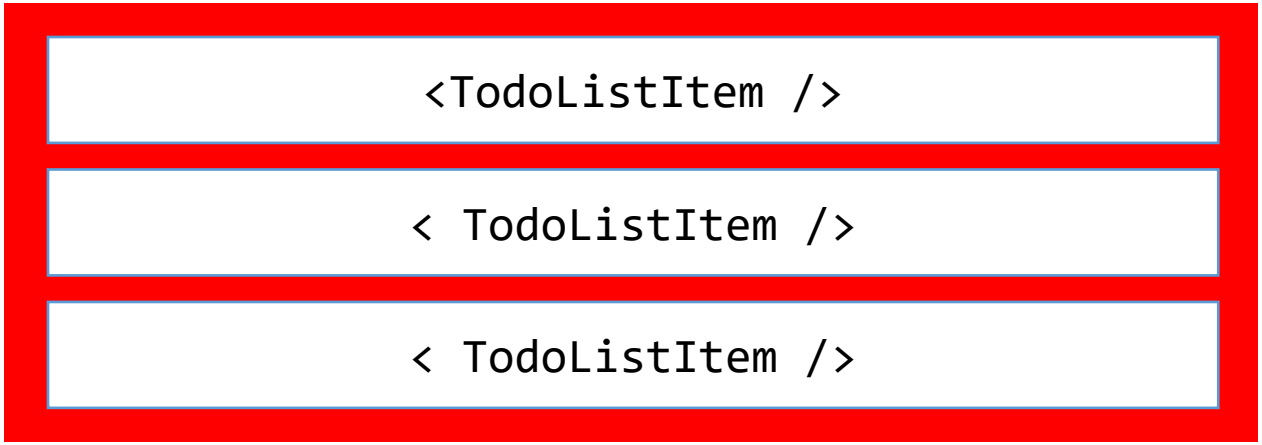


3. 리스트 만들기

App.js

```
<TodoTemplate>
      <TodoInsert/>
      <TodoList/>
</TodoTemplate>
```

TodoList.js



3. 리스트 만들기

❖ App.js

```
import './App.css';
import TodoTemplate from './components/TodoTemplate';
import TodoInsert from './components/TodoInsert';
import TodoList from './components/TodoList';

function App() {
  return (
    <TodoTemplate>
      <TodoInsert/>
      <TodoList/>
    </TodoTemplate>
  );
}

export default App;
```

3. 리스트 만들기

❖ TodoList.js

```
import React from 'react';
import TodoListItem from './TodoListItem';
import '../style/TodoList.scss';

const TodoList = () => {
    return (
        <div className="TodoList">
            <TodoListItem />
            <TodoListItem />
            <TodoListItem />
        </div>
    );
};

export default TodoList;
```

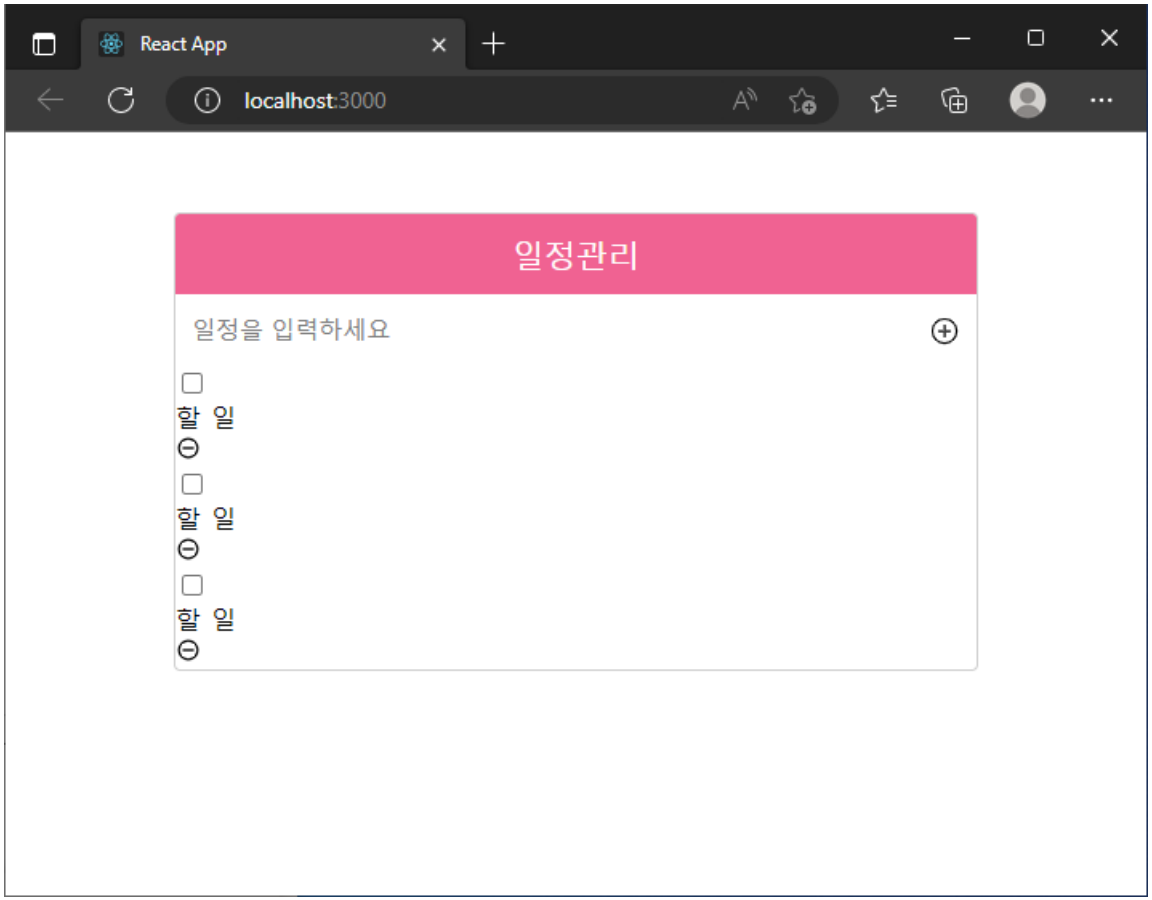
3. 리스트 만들기

❖ TodoListItem.js

```
import React from 'react';
import {IoIosCheckboxOutline, IoIosSquareOutline,
IoMdRemoveCircleOutline} from 'react-icons/io';
import '../style/TodoListItem.scss';

const TodoListItem = () => {
    return (
        <div className="TodoListItem">
            <div className="todo">
                <input type="checkbox" className="checkbox" />
                <div className="text">할 일</div>
            </div>
            <div className="remove">
                <IoMdRemoveCircleOutline />
            </div>
        </div>
    );
};

export default TodoListItem;
```

3. 리스트 만들기

❖ 실행 결과



3. 리스트 만들기

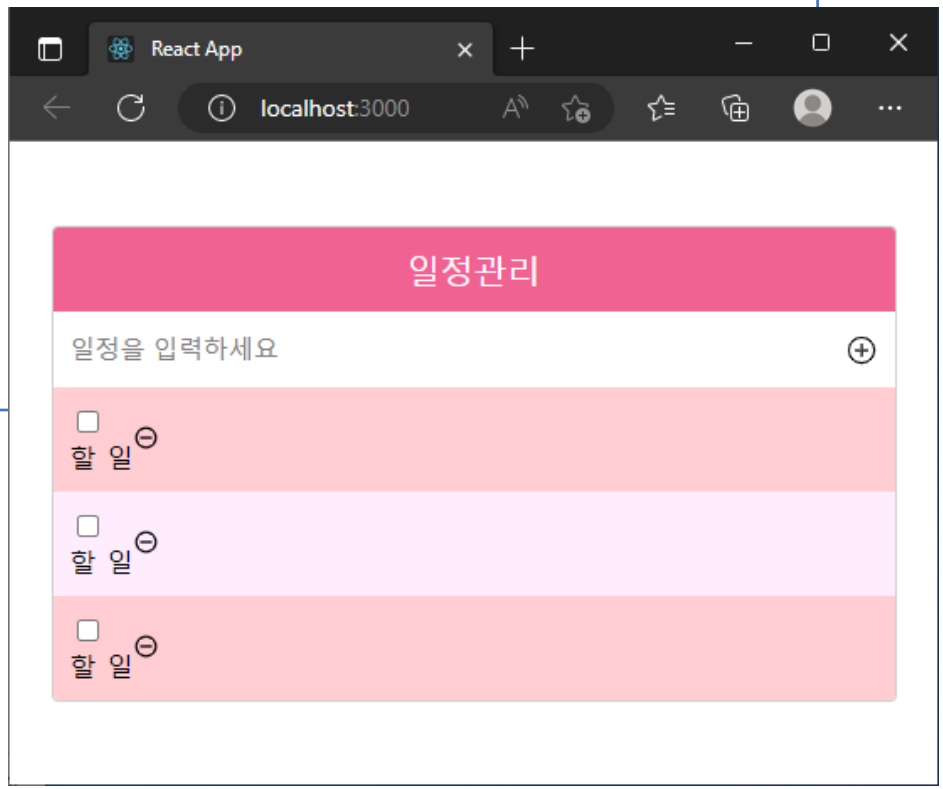
❖ TodoList.scss

```
.TodoList {
    min-height: 100px;
    max-height: 500px;

    // max-height 넘쳤을 때 스크롤 생성
    overflow-y : auto;
}
```

3. 리스트 만들기

❖ TodoListItem.scss

```
.TodoListItem {
    padding: 10px;
    display: flex;
    align-items: center;
    background: #ffcdd2;

    // 짝수번째 행 배경색 변경
    &:nth-child(even) {
        background: #ffedff;
    }
}
```



3. 리스트 만들기

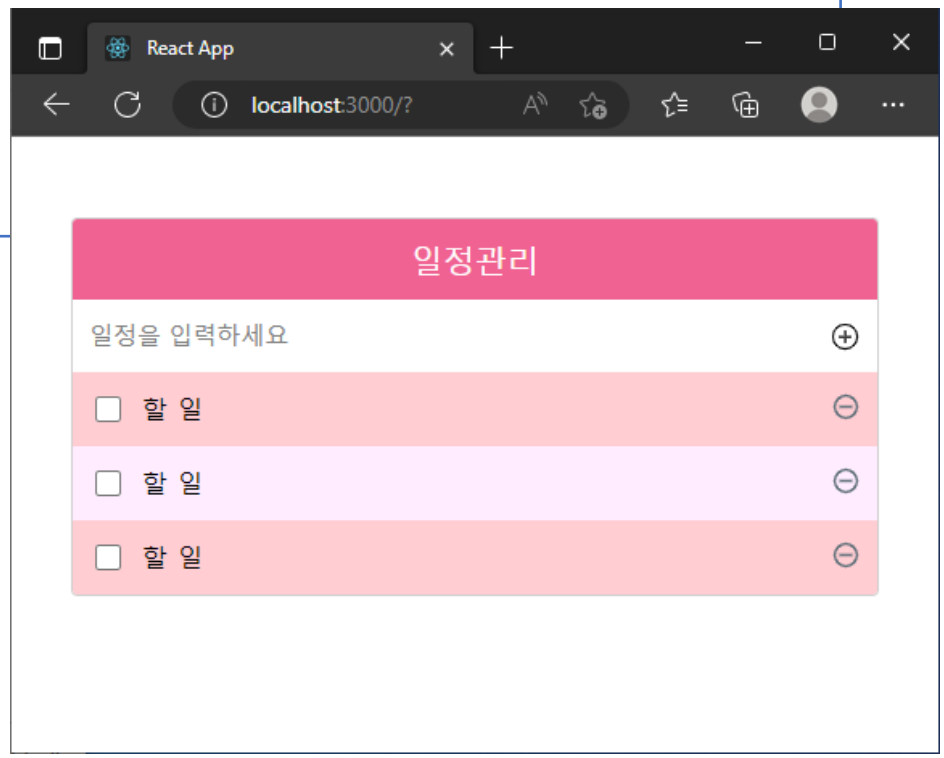
❖ TodoListItem.scss

```
.todo {
  display: flex;

  .checkbox {
    width: 1rem;
    height: 1rem;
  }

  .text {
    margin-left: 10px;
  }
}
```

3. 리스트 만들기

❖ TodoListItem.scss

```
.remove {
  color: #62727b;
  font-size: 1.2rem;
  cursor: pointer;

  &:hover {
    color: #37474f;
  }
}
```



03 기능 구현하기

기존 내용 값 전달하기

❖ **App.js**
- 추가될 내용 항목의 상태 관리
- useState를 사용하여 todos라는 상태를 정의
- todos를 TodoList의 props로 전달

기존 내용 값 전달하기

❖ App.js

```
import { useState } from 'react';

function App() {
  const [todos, setTodos]= useState(
    [
      {
        id : 1,
        text : '리액트 기초 공부하기',
        status : true
      },
      {
        id : 2,
        text : '포트폴리오 만들기',
        status : true
      },
      {
        id : 3,
        text : '프로젝트 준비하기',
        status : false
      }
    ]
  )
```

```
return (
    <TodoTemplate>
      <TodoInsert/>
      <TodoList todos={todos}/>
    </TodoTemplate>
  );
}
```

기존 내용 값 전달하기

❖ **TodoList.js**

```
import React from 'react';
import TodoListItem from './TodoListItem';
import '../style/TodoList.scss';

const TodoList = ( props ) => {
    const { todos } = props;

    return (
        <div className='TodoList’>
            <TodoListItem/>
            <TodoListItem/>
            <TodoListItem/>

            { todos.map(todo => <TodoListItem todo={todo}
                                              key={todo.id}/>)

            }
        </div>
    );
};

export default TodoList;
```

기존 내용 값 전달하기

❖ TodoListItem.js

```
import { IoMdRemoveCircleOutline } from "react-icons/io";
import "../style/TodoListItem.scss";

const TodoListItem = ( props ) => {
  const { todo } = props;

  return (
    <div className="TodoListItem">
       <div className="todo">
          <input type="checkbox" className="checkbox" />
          <div className="text">{todo.text}</div>
      </div>
      <div className="remove">
          <IoMdRemoveCircleOutline />
      </div>
    </div>
  );
};
export default TodoListItem;
```

기존 내용 값 전달하기

❖ 실행 결과

체크 박스 선택 시 밑줄 생성

❖ **TodoListItem.scss**

```
.todo {
  display: flex;

  .checkbox {
     width: 1rem;
     height: 1rem;

     &:checked + .text {
       color: grey;
       text-decoration: line-through;
     }
  }

  .text {
     margin-left: 10px;
     flex: 1 1 0;
  }
}
```

체크 박스 선택 시 밑줄 생성

❖ 실행 결과

항목 추가 기능 구현하기

❖ **App.js**

```
// 생략

const nextId = useRef(4);

return (
    <>
      <TodoTemplate>
        <TodoInsert insertItem={insertHandler} />
        <TodoList todos={todos} />
      </TodoTemplate>
    </>
);
```

항목 추가 기능 구현하기

❖ **App.js**

- 항목 추가 이벤트 리스너

```
const insertHandler = (item) => {
    // 새로운 todo 객체 생성
    const todo = {
      id: nextId.current,
      text: item,
      status: false,
    };

    // 기존에 있던 todos에 새로운 todo를 추가
    setTodos([...todos, todo]);

    nextId.current += 1;
};
```

항목 추가 기능 구현하기

❖ TodoInsert.js

```
const TodoInsert = ({ insertItem }) => {
    // 추가한 todo 아이템
    const [item, setItem] = useState("");

    return (
        <form className="TodoInsert" onSubmit={submitHandler}>
          <input type="text" placeholder="일정을 입력하세요"
                 value={item} onChange={changeHandler} />
          <button type="submit">
             <IoIosAddCircleOutline />
          </button>
        </form>
    );
};

export default TodoInsert;
```

항목 추가 기능 구현하기

❖ **TodoInsert.js**

- onChange 이벤트 설정

```
const changeHandler = (e) => setItem(e.target.value);
```

항목 추가 기능 구현하기

❖ **TodoInsert.js**

- onSubmit 이벤트 설정

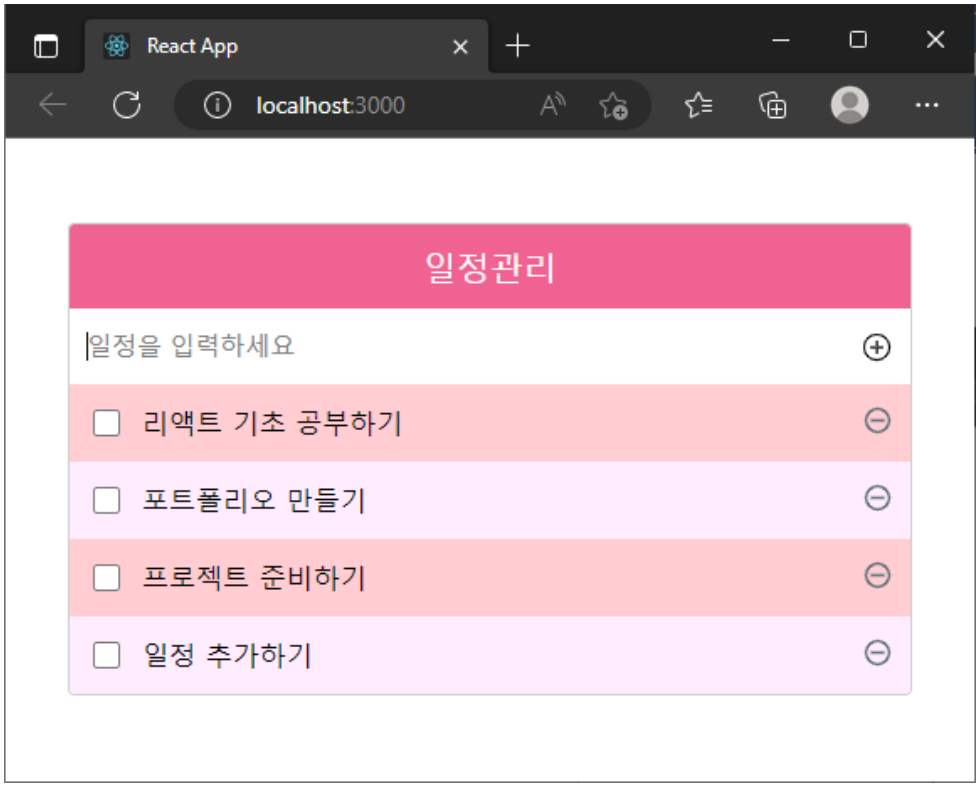
```
// submit 이벤트 처리
const submitHandler = (e) => {
    // 페이지 새로고침 방지
    e.preventDefault();

    // 빈칸을 입력한 경우, 추가 안함
    if (item.trim() === "") {
      return;
    }

    // 아이템 추가
    insertItem(item);

    // input 창에 텍스트 비우기
    setItem("");
};
```

항목 추가 기능 구현하기

❖ 실행 결과

항목 추가 기능 구현하기

❖ 실행 결과



항목 제거 기능 구현하기

❖ **항목 제거**

- todos 배열에서 id 값으로 항목을 제거
- filter() 함수를 이용

항목 제거 기능 구현하기

❖ **App.js**

```
// 생략

return (
    <>
      <TodoTemplate>
        <TodoInsert insertItem={insertHandler} />
        <TodoList todos={todos} removeItem={removeHandler} />
      </TodoTemplate>
    </>
);
```

항목 제거 기능 구현하기

❖ **App.js**
- 항목 제거 이벤트 리스너

```
const removeHandler = (deleted) =>
    setTodos(todos.filter((prevTodo) => prevTodo.id !== deleted));
```

항목 제거 기능 구현하기

❖ **TodoList.js**

```
import React from 'react';
import TodoListItem from './TodoListItem';
import '../style/TodoList.scss';

const TodoList = ( props ) => {
    const { todos, removeItem } = props;

    return (
             <div className="TodoList">
               {
                  todos.map(item => (
                                  <TodoListItem todo={item}
                                                key={item.id}
                                                removeItem={removeItem}/>))
               }
             </div>
           );
};

export default TodoList;
```

항목 제거 기능 구현하기

❖ TodoListItem.js

```
const TodoListItem = ( props ) => {
  const { todo, removeItem } = props;

  return (
    <div className="TodoListItem">
      // 생략

      <div className="remove" onClick={() => removeItem(todo.id)}>
         <IoMdRemoveCircleOutline />
      </div>
    </div>
  );
};
```

항목 제거 기능 구현하기

❖ 실행 결과

항목 수정 기능 구현하기

❖ **App.js**

```
// 생략

return (
    <>
      <TodoTemplate>
        <TodoInsert insertItem={insertHandler} />
        <TodoList todos={todos} removeItem={removeHandler}
                                updateItem={updateHandler} />
      </TodoTemplate>
    </>
);
```

항목 수정 기능 구현하기

❖ **App.js**

- 항목 추가 이벤트 리스너

```
const updateHandler = (updated) => setTodos(
      todos.map(prevTodo =>
                  prevTodo.id === updated.id ? updated : prevTodo)
);
```

항목 수정 기능 구현하기

❖ **TodoList.js**

```
import React from 'react';
import TodoListItem from './TodoListItem';
import '../style/TodoList.scss';

const TodoList = ( props ) => {
    const { todos, removeItem, updateItem } = props;

    return (
            <div className="TodoList">
              {
                 todos.map(item => (
                                <TodoListItem todo={item}
                                              key={item.id}
                                              removeItem={removeItem}
                                              updateItem={updateItem}/>))
              }
            </div>
    );
};

export default TodoList;
```

항목 수정 기능 구현하기

❖ TodoListItem.js

```
const TodoListItem = (props) => {
  const { todo, removeItem, updateItem } = props;

  return (
    <div className="TodoListItem">
    <div className="todo">
       <input type="checkbox" className="checkbox"
              checked={todo.status === true}
              onChange={changeHandler}
        />
       <div className="text">{todo.text}</div>
      </div>

    // 생략

    </div>
  );
};
```

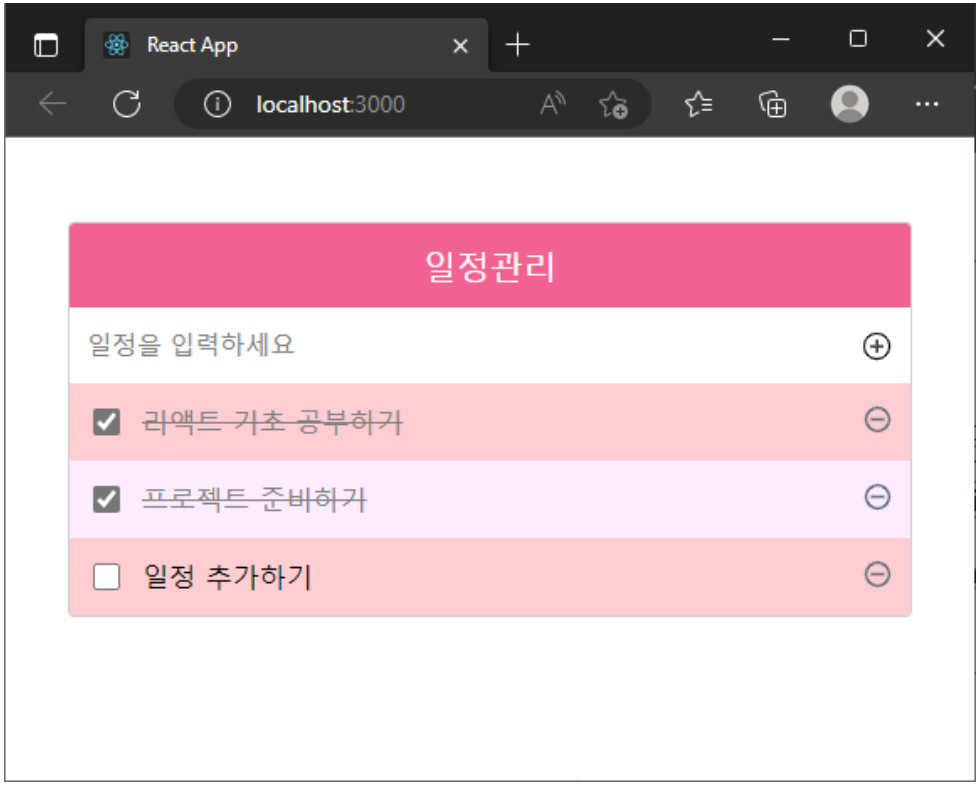
항목 수정 기능 구현하기

❖ TodoListItem.js

```
const TodoListItem = (props) => {
  const { todo, removeItem, updateItem } = props;

  const changeHandler = (e) => {
     const checked = e.target.checked ? true : false;
     updateItem({ ...todo, status: checked });
  };

  return (
    <div className="TodoListItem">
      // 생략
    </div>
  );
};
```

항목 수정 기능 구현하기

❖ 실행 결과




THANK YOU

Props Drilling

❖ Props Drilling