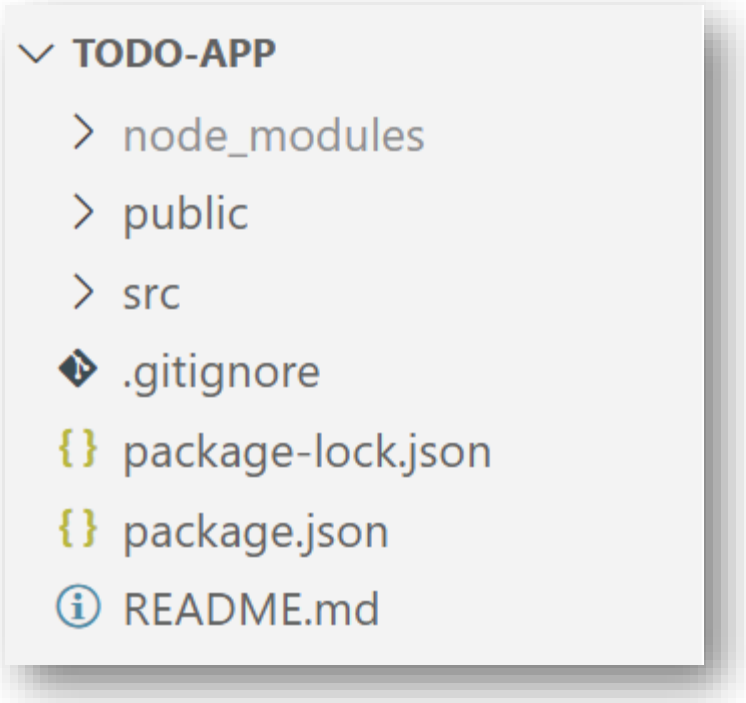
일정 관리 프로그램 만들기

1. 프로젝트 생성
2. 컴포넌트 및 스타일 작성
3. 일정 등록 기능
4. 일정 삭제 기능
5. 일정 수정 기능
6. 성능 최적화

■ 프로젝트 생성

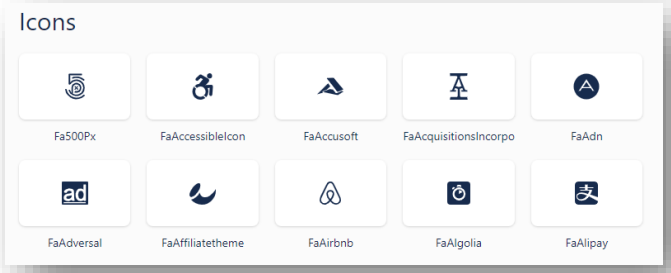
● 프로젝트명 : todo-app



컴포넌트 구성

1. TodoTemplate : 기본 화면
2. TodoInsert : 새로운 항목 추가
3. TodoListItem : 각 항목에 대한 상세 정보
4. TodoList : 항목 리스트 출력

● 라이브러리 추가 설치 : npm install react-icons



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

■ 컴포넌트 및 스타일 작성 - 메인 화면

● 기본 Template

App.js

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

function App() {
  return (
    <TodoTemplate>Todo App을 만들자</TodoTemplate>
  );
}

export default App;
```

index.css

```css
body {
  margin: 0;
  padding: 0;
  background: #e9ecef;
}
```

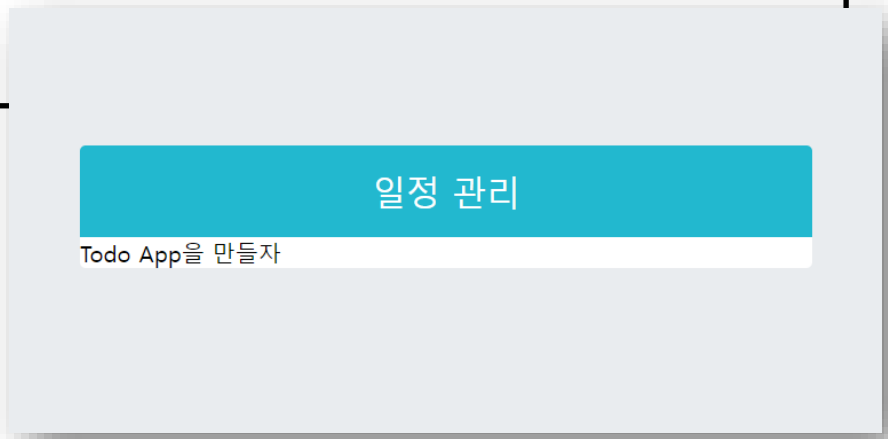
■ 컴포넌트 및 스타일 작성 - 메인 화면

● 기본 Template

components/TodoTemplate.js

```
import React from 'react';
import './TodoTemplate.css';

const TodoTemplate = ({ children }) => {
  return (
    <div className="TodoTemplate">
      <div className="app-title">일정 관리</div>
      <div className="content">{children}</div>
    </div>
  );
};

export default TodoTemplate;
```

부모 컴포넌트에서
현재 컴포넌트를 호출하면서 태그 사이에 넣은 문자열
ex) <OOO>*문자열*</OOO>

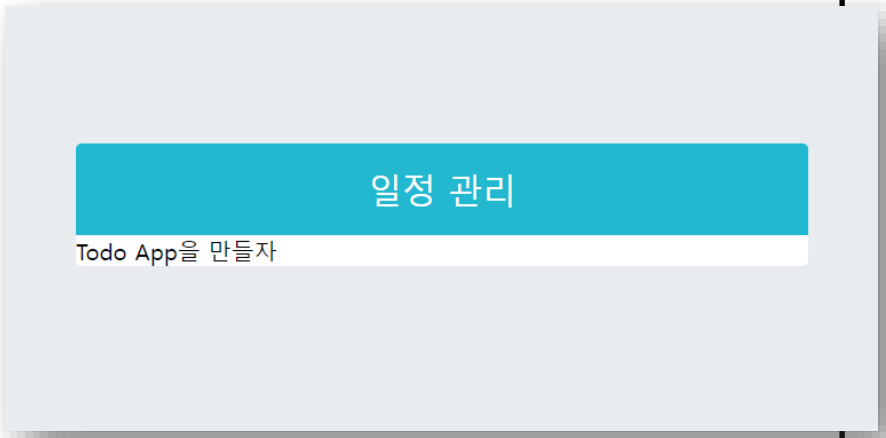


■ 컴포넌트 및 스타일 작성 - 메인 화면

● 기본 Template

components/TodoTemplate.css

```css
.TodoTemplate {
  width: 512px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 6rem;
  border-radius: 4px;
  overflow: hidden;
}
.TodoTemplate .app-title {
  background: #22b8cf;
  color: white;
  height: 4rem;
  font-size: 1.5rem;
  display: flex;
  align-items: center;
  justify-content: center;
}
.TodoTemplate .content {
  background: white;
}
```

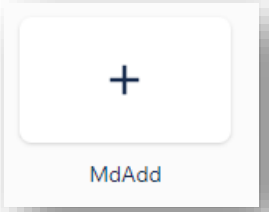


■ 컴포넌트 및 스타일 작성 - 일정 등록

● TodoInsert

components/TodoInsert.js

```
import { MdAdd } from 'react-icons/md';
import './TodoInsert.css';

const TodoInsert = () => {
  return (
    <form className="TodoInsert">
      <input placeholder="할 일을 입력하세요" />
      <button type="submit">
        <MdAdd />
      </button>
    </form>
  );
};

export default TodoInsert;
```

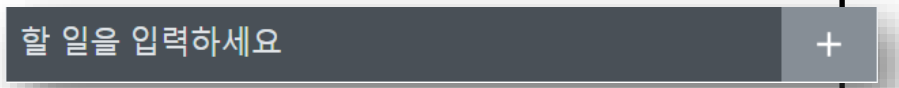


■ 컴포넌트 및 스타일 작성 - 일정 등록

● TodoInsert

components/TodoInsert.css

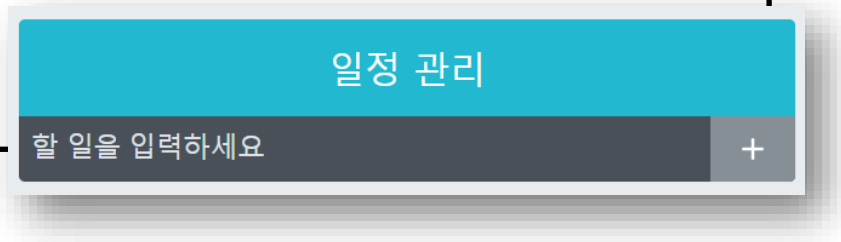
```css
.TodoInsert { display: flex; background: #495057; }

.TodoInsert input {
  background: none; outline: none; border: none;
  padding: 0.5rem; font-size: 1.125rem; line-height: 1.5;
}

.TodoInsert input::placeholder { color: #dee2e6; }

.TodoInsert button {
  background: none; outline: none; border: none; cursor: pointer;
  background: #868e96;
  color: white; display: flex;
  padding-left: 1rem;
  padding-right: 1rem;
  font-size: 1.5rem;
  align-items: center;
  transition: 0.1s background ease-in;
}

.TodoInsert button:hover { background: #adb5bd; }
```



■ 컴포넌트 및 스타일 작성 - 일정 등록

App.js

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

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

export default App;
```

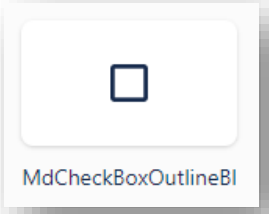


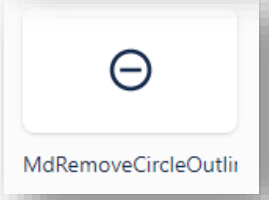
■ 컴포넌트 및 스타일 작성 - 일정 목록

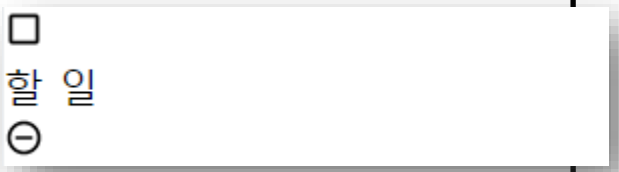
● 항목

components/TodoListItem.js

```
import React from 'react';
import {
  MdCheckBoxOutlineBlank, MdRemoveCircleOutline,
} from 'react-icons/md';
import './TodoListItem.css';

const TodoListItem = () => {
  return (
    <div className="TodoListItem">
      <div className='checkbox'>
        <MdCheckBoxOutlineBlank></MdCheckBoxOutlineBlank>
        <div className="text">할 일</div>
      </div>
      <div className="remove">
        <MdRemoveCircleOutline></MdRemoveCircleOutline>
      </div>
    </div>
  );
};

export default TodoListItem;
```





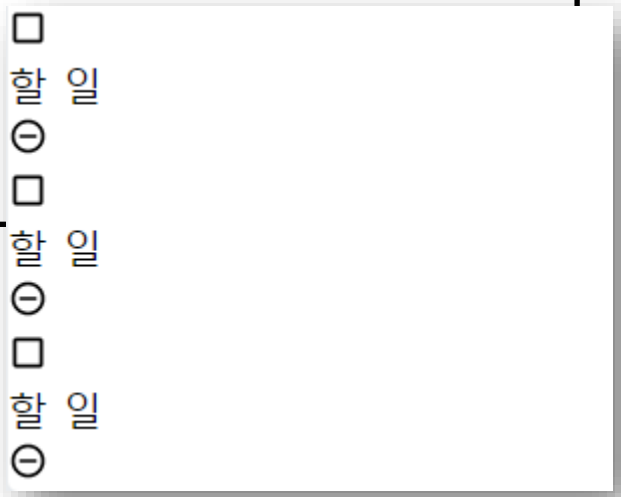


■ 컴포넌트 및 스타일 작성 - 일정 목록

● 목록

components/TodoList.js

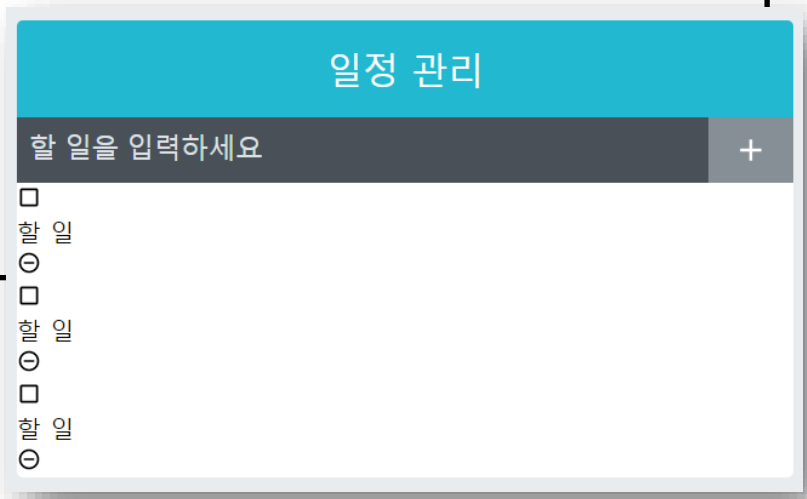
```
import TodoListItem from './TodoListItem';
import './TodoList.css';

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

export default TodoList;
```



■ 컴포넌트 및 스타일 작성 - 일정 목록

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></TodoInsert>
      <TodoList></TodoList>
    </TodoTemplate>
  );
}

export default App;
```

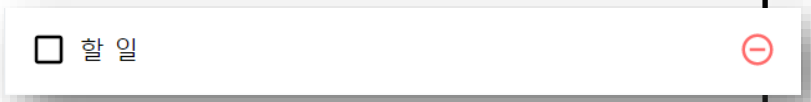


■ 컴포넌트 및 스타일 작성 - 일정 목록

● 항목 스타일

components/TodoListItem.css

```css
.TodoListItem { padding: 1rem; display: flex; align-items: center; }

.TodoListItem:nth-child(even) { background: #f8f9fa; }

.TodoListItem .checkbox {
  cursor: pointer; flex: 1; display: flex; align-items: center;
}
.TodoListItem .checkbox svg { font-size: 1.5rem; }
.TodoListItem .checkbox .text { margin-left: 0.5rem; flex: 1; }
.TodoListItem .checkbox.checked svg { color: #22b8cf; }
.TodoListItem .checkbox.checked .text {
  color: #adb5bd; text-decoration: line-through;
}

.TodoListItem .remove {
  display: flex; align-items: center;
  font-size: 1.5rem; color: #ff6b6b; cursor: pointer;
}
.TodoListItem .remove:hover { color: #ff8787; }

.TodoListItem + .TodoListItem { border-top: 1px solid #dee2e6; }
```

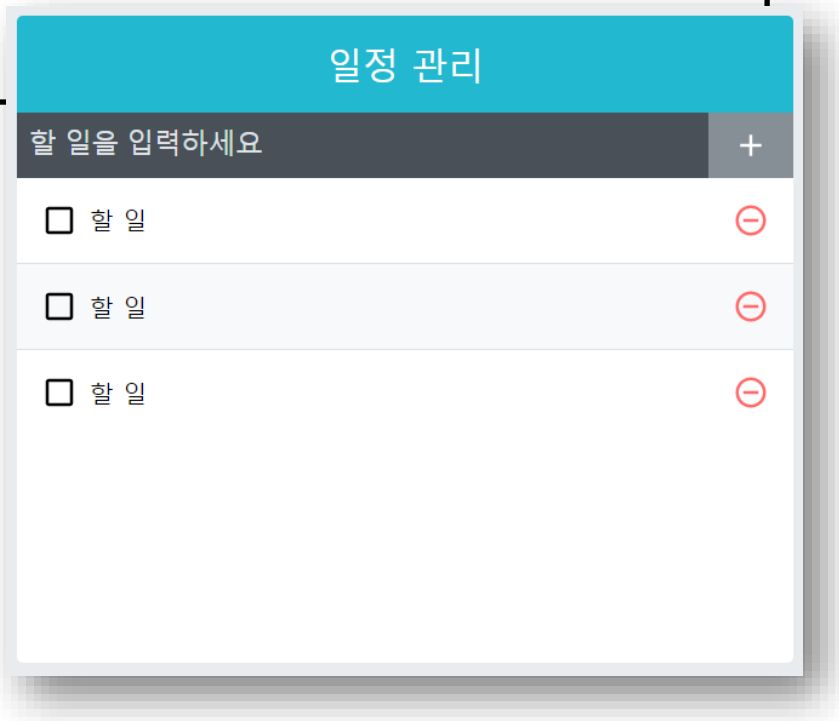


■ 컴포넌트 및 스타일 작성 - 일정 목록

● 목록 스타일

components/TodoList.css

```css
.TodoList {
  min-height: 320px;
  max-height: 513px;
  overflow-y: auto;
}
```



■ 일정 등록 기능 구현

● props로 전달된 todos 데이터를 TodoItem으로 다시 전달

components/TodoList.js

```
import TodoListItem from './TodoListItem';
import './TodoList.css';

const TodoList = (props) => {
  const list = [];
  for(let i = 0; i < props.todos.length; i++) {
    list.push(
      <TodoListItem todo={props.todos[i]} key={props.todos[i].id} />
    );
  }
  return (
    <div className="TodoList">
      { list }
    </div>
  );
};

export default TodoList;
```

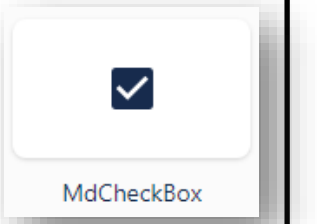
```
props.todos.map((todo) => {
  return <TodoListItem todo={todo} key={todo.id} />
})
```

■ 일정 등록 기능 구현

● todo의 checked에 따라 항목의 모습을 다르게 출력

components/TodoListItem.js

```
...

import {
  MdCheckBoxOutlineBlank,
  MdRemoveCircleOutline,
  MdCheckBox
} from 'react-icons/md';

const TodoListItem = (props) => {
  const { text, checked } = props.todo;
  return (
    <div className="TodoListItem">
      <div className={checked ? 'checkbox checked' : 'checkbox'}>
        {checked ? <MdCheckBox /> : <MdCheckBoxOutlineBlank />}
        <div className="text">{text}</div>
      </div>

      ...
```

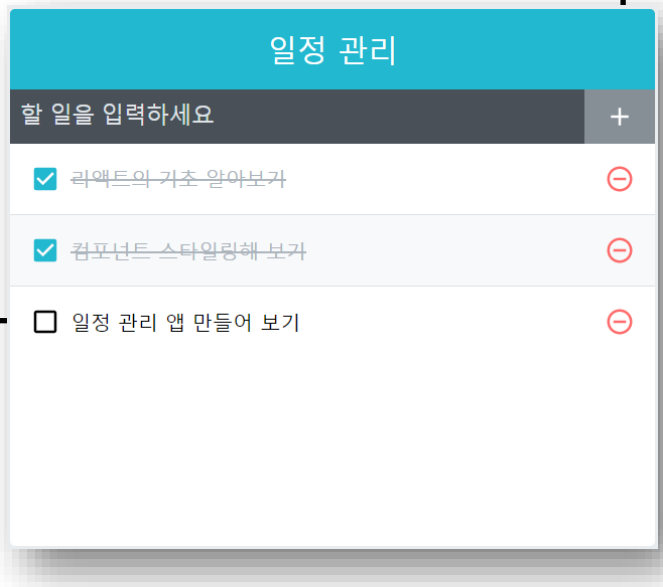


■ 일정 등록 기능 구현

● App의 state로 데이터를 정의하고 TodoList에 props로 전달

App.js

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

function App() {
  const [todos, setTodos] = useState([
    { id: 1, text: '리액트의 기초 알아보기', checked: true },
    { id: 2, text: '컴포넌트 스타일링해 보기', checked: true },
    { id: 3, text: '일정 관리 앱 만들어 보기', checked: false }
  ]);

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

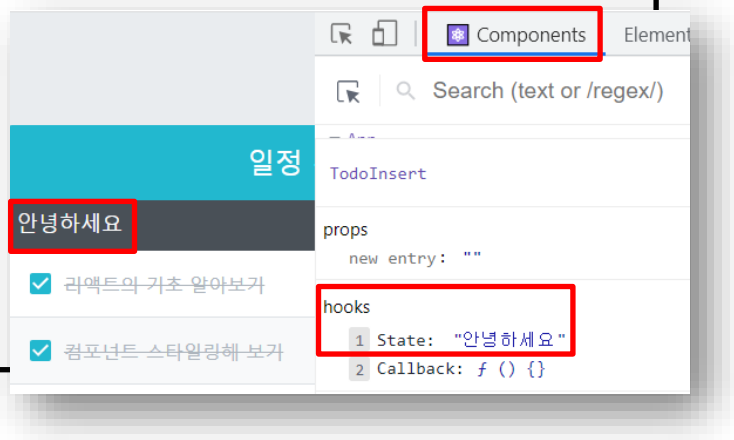


■ 일정 등록 기능 구현

● TodoInsert의 value 상태 관리

components/TodoInsert.js

```
...
import { useState, useCallback } from 'react';

const TodoInsert = () => {
  const [value, setValue] = useState('');

  const onChange = useCallback((e) => {
    setValue(e.target.value);  컴포넌트가 렌더링 될 때마다 생성하지 않고 재사용
  });

  return (
    <form className="TodoInsert">
      <input placeholder="할 일을 입력하세요" value={value}
        onChange={onChange} />
      ...
    </form>
  );
};

export default TodoInsert;
```



■ 일정 등록 기능 구현

● App의 todos 배열에 항목 추가하기

App.js

```
import { useState, useRef, useCallback } from 'react';

function App() {
  ...
  const nextId = useRef(todos.length + 1);
  const onInsert = useCallback(
    (text) => {
      const todo = {
        id: nextId.current, text,
        checked: false,
      };
      setTodos(todos.concat(todo));
      nextId.current += 1;
    },
    [todos]
  );

  return (
    <TodoTemplate>
      <TodoInsert onInsert={onInsert}></TodoInsert>
      ...
```

useRef : 고유한 값을 저장 할 때 사용
id는 현재 상태를 유지해야 하는 고유한 값이고 화면 출력용이 아님

※ todos.push() 사용 불가

useCallback 함수 안에서 사용하는 state 또는 props가 있다면
deps 배열로 반드시 지정해야 최신 값을 보장할 수 있음

■ 일정 등록 기능 구현

● App의 todos 배열에 항목 추가하기

components/TodoInsert.js

```
...
import { useState, useCallback } from 'react';

const TodoInsert = (props) => {
  const [value, setValue] = useState('');

  const onChange = useCallback((e) => {
    setValue(e.target.value);
  });

  const onSubmit = useCallback((e) => {
    props.onInsert(value);
    setValue('');
    e.preventDefault();
  }, [value]);

  return (
    <form className="TodoInsert" onSubmit={onSubmit}>
      <input placeholder="할 일을 입력하세요" value={value}
        onChange={onChange} />
      ...
```

■ 일정 삭제 기능 구현

● App에서 onRemove 함수 작성 후 TodoList에 props로 전달

App.js

```
import { useState, useRef, useCallback } from 'react';

function App() {
  ...
  const onRemove = useCallback(
    (id) => {
      setTodos(todos.filter((todo) => todo.id !== id));
    },
    [todos]
  );

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

■ 일정 삭제 기능 구현

● TodoList는 TodoListItem으로 props를 그대로 전달

components/TodoList.js

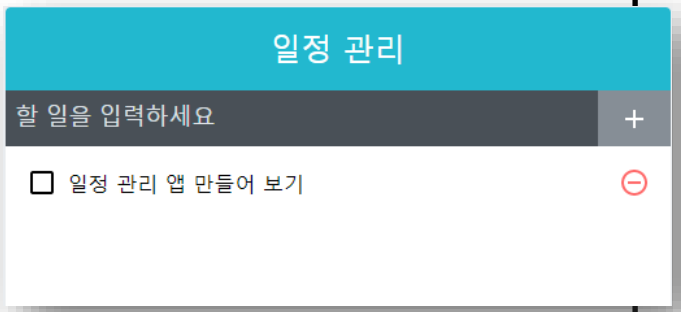
```
import TodoListItem from './TodoListItem';
import './TodoList.css';

const TodoList = (props) => {
  const list = [];
  for(let i = 0; i < props.todos.length; i++) {
    list.push(
      <TodoListItem todo={props.todos[i]} key={props.todos[i].id}
          onRemove={props.onRemove} />
    );
  }
  return (
    <div className="TodoList">
      { list }
    </div>
  );
};

export default TodoList;
```

■ 일정 삭제 기능 구현

● TodoListItem이 가지고 있는 todo의 id로 onRemove 함수 호출

components/TodoListItem.js

```
...

import {
  MdCheckBoxOutlineBlank,
  MdRemoveCircleOutline,
  MdCheckBox
} from 'react-icons/md';

const TodoListItem = (props) => {
  const { id, text, checked } = props.todo;
  return (
    <div className="TodoListItem">
      ...

      <div className="remove" onClick={() => {
        props.onRemove(id);
      }}>
        <MdRemoveCircleOutline></MdRemoveCircleOutline>
      </div>

      ...
```



■ 일정 수정 기능 구현

● App에서 onToggle 함수 작성 후 TodoList에 props로 전달

App.js

```
function App() {
  ...
  const onToggle = useCallback(
    (id) => {
      setTodos(
        todos.map((todo) =>
          todo.id === id ? { ...todo, checked: !todo.checked } : todo
        ),
      );
    },
    [todos]
  );

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

■ 일정 수정 기능 구현

● TodoList는 TodoListItem으로 props를 그대로 전달

components/TodoList.js

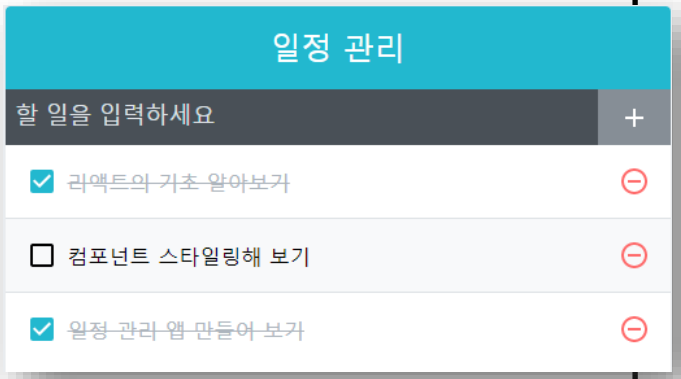
```
import TodoListItem from './TodoListItem';
import './TodoList.css';

const TodoList = (props) => {
  const list = [];
  for(let i = 0; i < props.todos.length; i++) {
    list.push(
      <TodoListItem todo={props.todos[i]} key={props.todos[i].id}
          onRemove={props.onRemove} onToggle={props.onToggle} />
    );
  }
  return (
    <div className="TodoList">
      { list }
    </div>
  );
};

export default TodoList;
```

■ 일정 수정 기능 구현

● TodoListItem이 가지고 있는 todo의 id로 onToggle 함수 호출

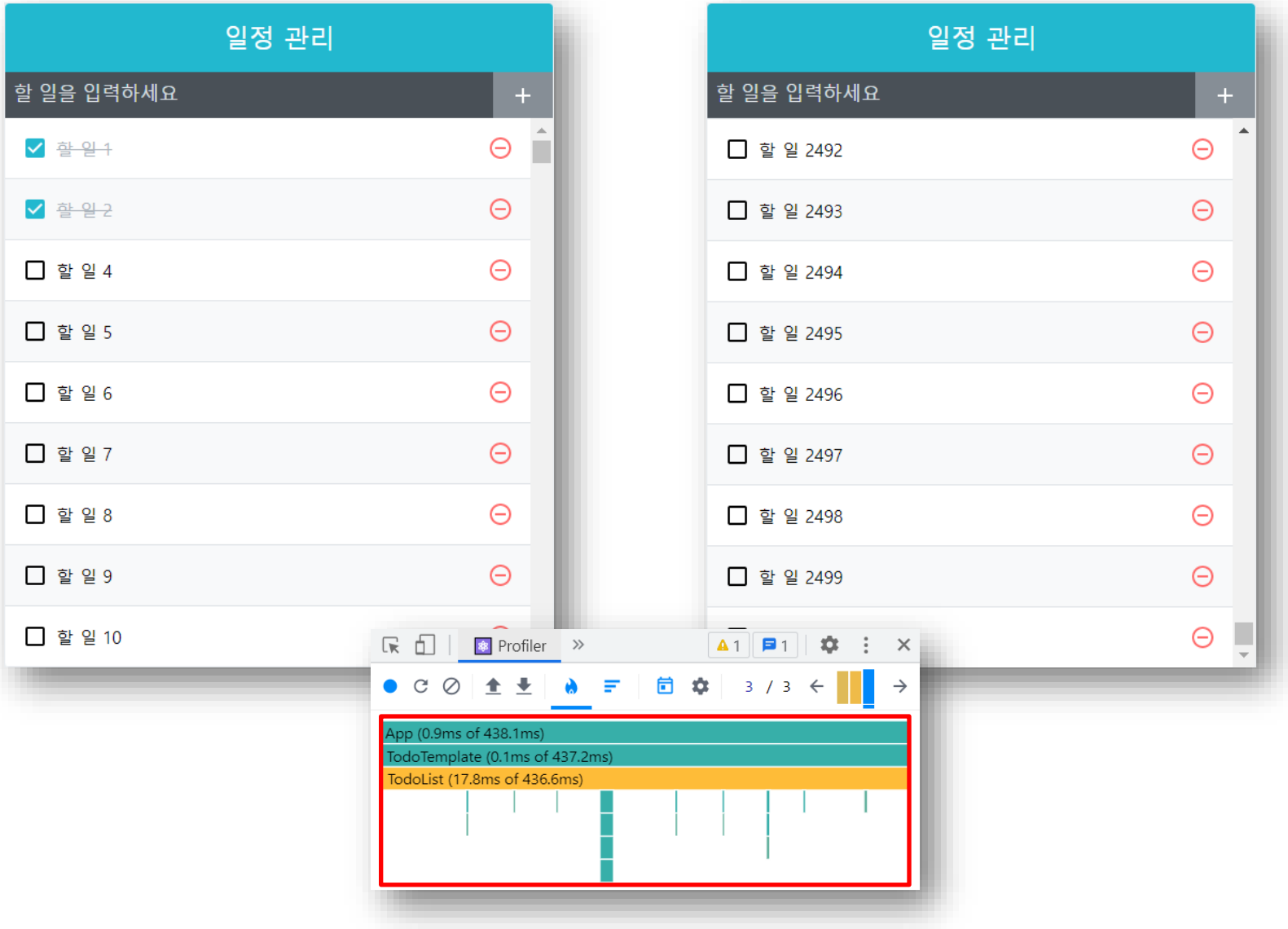
components/TodoListItem.js

```
...

import {
  MdCheckBoxOutlineBlank,
  MdRemoveCircleOutline,
  MdCheckBox
} from 'react-icons/md';

const TodoListItem = (props) => {
  const { id, text, checked } = props.todo;
  return (
    <div className="TodoListItem">
      <div className={checked ? 'checkbox checked' : 'checkbox'}
          onClick={() => props.onToggle(id)}>
        {checked ? <MdCheckBox /> : <MdCheckBoxOutlineBlank />}
        <div className="text">{text}</div>
      </div>
      ...
```



■ 많은 데이터 렌더링하기

- 데이터 2,500개 입력



■ 많은 데이터 렌더링하기

● 컴포넌트 리렌더링 발생 원인

1. 전달받은 props가 변경될 때

2. state가 변경될 때

3. 부모 컴포넌트가 리렌더링될 때

4. forceUpdate 함수가 실행될 때

➔ 항목을 체크하거나 입력, 삭제하는 경우

App 컴포넌트의 state 변경(리렌더링) → TodoList → TodoListItem

● 해결 방법

- 클래스 컴포넌트 : shouldComponentUpdate 함수 사용
- 함수 컴포넌트 : **React.memo 함수** + **useState의 함수형 업데이트** 방식

■ 많은 데이터 렌더링하기

● 함수형 업데이트 방식으로 코드 수정

App.js (onInsert)

```
...

const onInsert = useCallback(
  (text) => {
    const todo = {
      id: nextId.current, text,
      checked: false,
    };

    setTodos(
      todos.concat(todo)
    );

    nextId.current += 1;
  },
  [todos]
);

...
```

```
...

const onInsert = useCallback(
  (text) => {
    const todo = {
      id: nextId.current, text,
      checked: false,
    };

    setTodos(
      todos => todos.concat(todo)
    );

    nextId.current += 1;
  },
  []
);

...
```

■ 많은 데이터 렌더링하기

● 함수형 업데이트 방식으로 코드 수정

App.js (onRemove)

```
...

const onRemove = useCallback(
  (id) => {
    setTodos(
      todos.filter(
        (todo) => todo.id !== id
      )
    );
  },
  [todos]
);


...
```

```
...

const onRemove = useCallback(
  (id) => {
    setTodos(
      todos.filter(todos =>
        (todo) => todo.id !== id
      )
    );
  },
  []
);


...
```

■ 많은 데이터 렌더링하기

● 함수형 업데이트 방식으로 코드 수정

App.js (onToggle)

```
...

const onToggle = useCallback(
  (id) => {
    setTodos(
      todos.map((todo) =>
        todo.id === id ?
          { ...todo,
          checked: !todo.checked } :
          todo
      )
    );
  },
  [todos]
);


...
```

```
...

const onToggle = useCallback(
  (id) => {
    setTodos(todos =>
      todos.map((todo) =>
        todo.id === id ?
          { ...todo,
          checked: !todo.checked } :
          todo
      )
    );
  },
  []
);


...
```

■ 많은 데이터 렌더링하기

● React.memo 함수 적용

components/TodoListItem.js

```
import React from 'react';
...

const TodoListItem = (props) => {
  const { id, text, checked } = props.todo;

  return (
    <div className="TodoListItem">
      <div className={checked ? 'checkbox checked' : 'checkbox'}
          onClick={() => props.onToggle(id)}>
        {checked ? <MdCheckBox /> : <MdCheckBoxOutlineBlank />}
        <div className="text">{text}</div>
      </div>
      ...
    </div>
  );
};

export default React.memo(TodoListItem);
```

이전 props와 현재 props를 비교하여 변경사항이 있을 때만 리렌더링

■ 많은 데이터 렌더링하기

● 성능 최적화 이후 로딩 시간 확인