10. 일정관리 애플리케이션

10.1 프로젝트 준비하기

10.1.1 프로젝트생성 및 필요 라이브러리 설치

> - 신규 : 10.Todo> `yarn create react-app .` : 모두 삭제하고 완전 새로운 app생성
> - 설치 : `yarn add sass classnames react-icons`
> - react-icons : https://react-icons.netlify.com
>   - SVG형태로 제공, 리액트컴퍼넌트처럼 매우 쉽게 사용가능
>   - 아이콘의 크기나 색상은 props 또는 CSS 스타일오 변경하여 사용할 수 있다.

10.1.2 prettier 설정

- 프로젝트 최상위 폴더에 .prettier파일을 생성

.prettierrc

```
{
  "singleQuote": true,
  "semi": true,
  "useTabs": false,
  "tabWidth": 2,
  "trailingComma": "all",
  "printWidth": 80
}
```

10.1.3 index.css 수정

index.css

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

10.1.4 App컴퍼넌트 초기화

App.js

```
const App = () => {
  return <div>Todo App 만들기!!!</div>
}

export default App;
```

10.2 UI 구성하기

1. TodoTempate : 화면가운데정렬, 앱타이틀(일정관리), children으로 내부 JSX를 props를 전달 받아서 랜더링
2. TodoInsert: 신규항목입력/추가, state를 통해 input상태관리
3. TodoListItem : 항목정보조회, todo객체를 props로 전달받아 state에 따라 다른 스타일의 UI를 제공
4. TodoList : todo배열을 props로 받아온 후, map을 사용해서 보수가ㅐ의 TodoListItem으로 변환

10.2.1 Todo Template

1. src\components 폴더생성 및 파일생성
    - TodoTemplate.js
    - TodoTemplate.scss

components/TodoTemplate.js

```
import './TodoTemplate.scss'

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

export default TodoTemplate
```

App.js

```
import TodoTemplate from "./components/TodoTemplate";

const App = () => {
  return <TodoTemplate>Todo App 만들기!!!</TodoTemplate>;
};

export default App;
```

- 컴퍼넌트가 다른 탭에서 열려 있지 않으면 자동완성이 되지 않는다. 닫혀 있는 파일에서 자동완성이 작동하려면
- `최상위 폴더에 jsconfig.json 파일이 있어야 한다.`

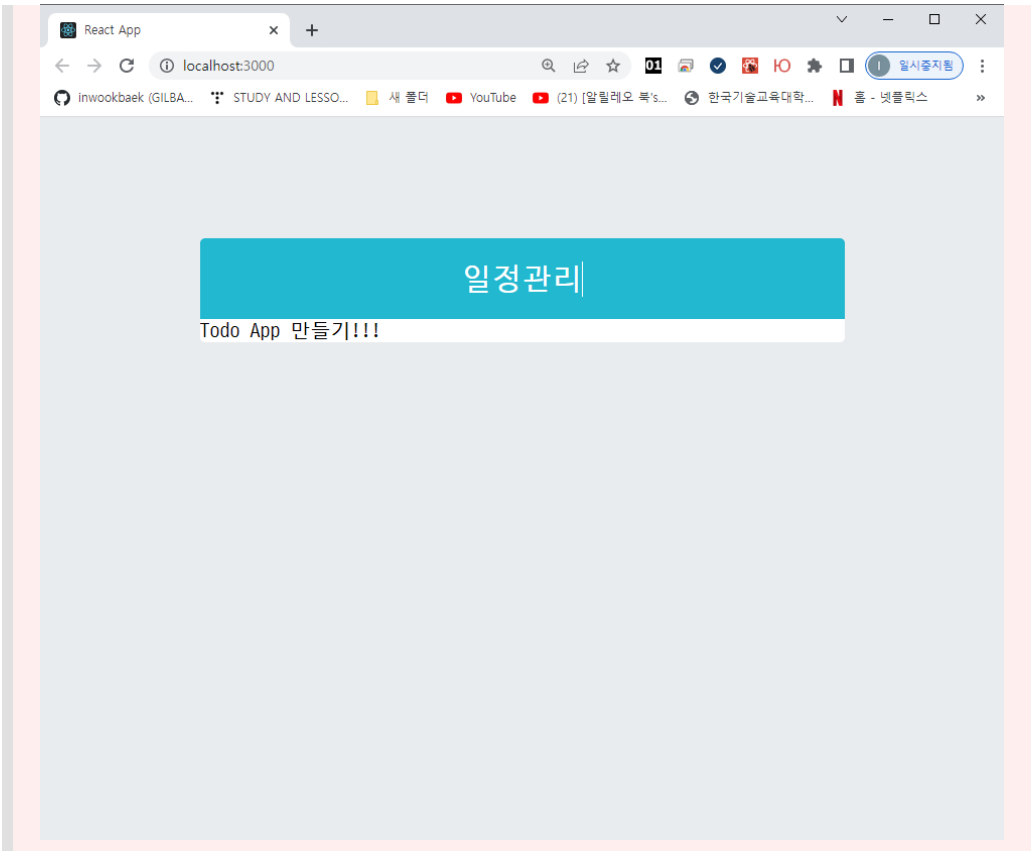
jsconfig.json

```
{
  "compilerOptions": {
    "target": "ES6"
  }
}
```

components/TodoTemplate.scss

```
.TodoTemplate {
  width: 512px;
  margin-left: auto; // width가 주어진 상태에서 좌우중앙정렬
  margin-right: auto;
```

```
    margin-top: 6rem;
    border-radius: 4px;
    overflow: hidden;

    .app-title {
      background: #22b8cf;
      color: white;
      height: 4rem;
      font-size: 1.5rem;
      display: flex;
      justify-content: center;
      align-items: center;
    }

    .content {
      background: white;
    }
  }
```



10.2.2 components/TodoInsert

TodoInsert.js

- MdAdd : https://react-icons.github.io/react-icons/icons?name=md

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


const TodoInsert = () => {
  return (
    <form className='TodoInsert'>
      <input placeholder='할 일을 입력하세요!!' />
```

```
        <button type='button'>
          <MdAdd />
        </button>
      </form>
    )
  }

  export default TodoInsert
```

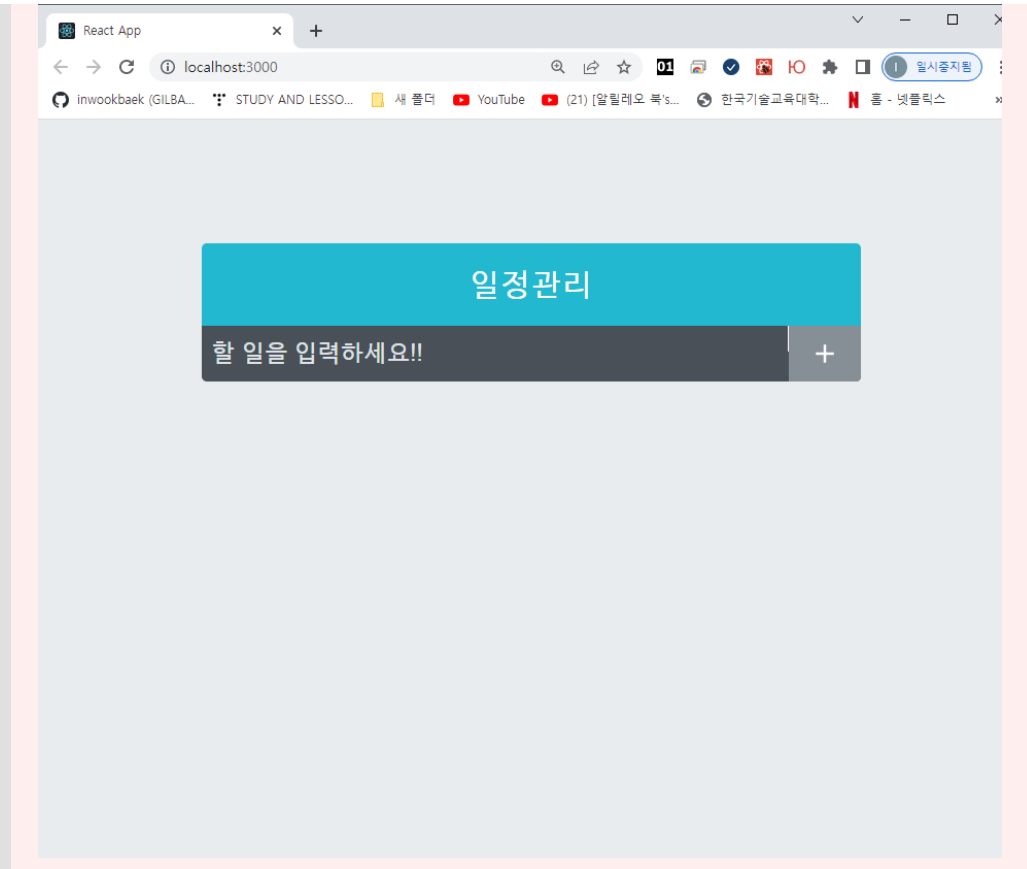
App.js

```
  import TodoInsert from "./components/TodoInsert";
  import TodoTemplate from "./components/TodoTemplate";

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

  export default App;
```

TodoInsert.scss

```
  .TodoInsert {
    display: flex;
    background: #495057;
    input {
      // 기본 스타일 초기화
      background: none;
      outline: none;
      border: none;
      padding: 0.5rem;
      font-size: 1.125rem;
      line-height: 1.5;
      color: white;
      &::placeholder {
        color: #dee2e6;
      }
      // 버튼을 제외한 영역을 모두 차지하기
      flex: 1;
    }
    button {
      // 기본 스타일 초기화
      background: none;
      outline: none;
      border: none;
      background: #868e96;
      color: white;
      padding-left: 1rem;
      padding-right: 1rem;
      font-size: 1.5rem;
      display: flex;
      align-items: center;
      cursor: pointer;
      transition: 0.1s background ease-in;
```

```
    &:hover {
      background: #adb5bd;
    }
  }
}
```



10.2.3 TodoListItem과 TodoList 만들기

TodoListItem.js

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

const TodoListItem = () => {

  return (
    <div className="TodoListItem">
      <div className='checkbox'>
        <MdCheckBoxOutlineBlank />
        <div className="text">오늘의 할일</div>
      </div>
      <div className="remove">
        <MdRemoveCircleOutline />
      </div>
    </div>
  );
};

export default TodoListItem;
```

TodoList.js

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

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

export default TodoList;
```

App.js

```
import {
import TodoInsert from "./components/TodoInsert";
import TodoList from "./components/TodoList";
import TodoTemplate from "./components/TodoTemplate";

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

export default App;
```

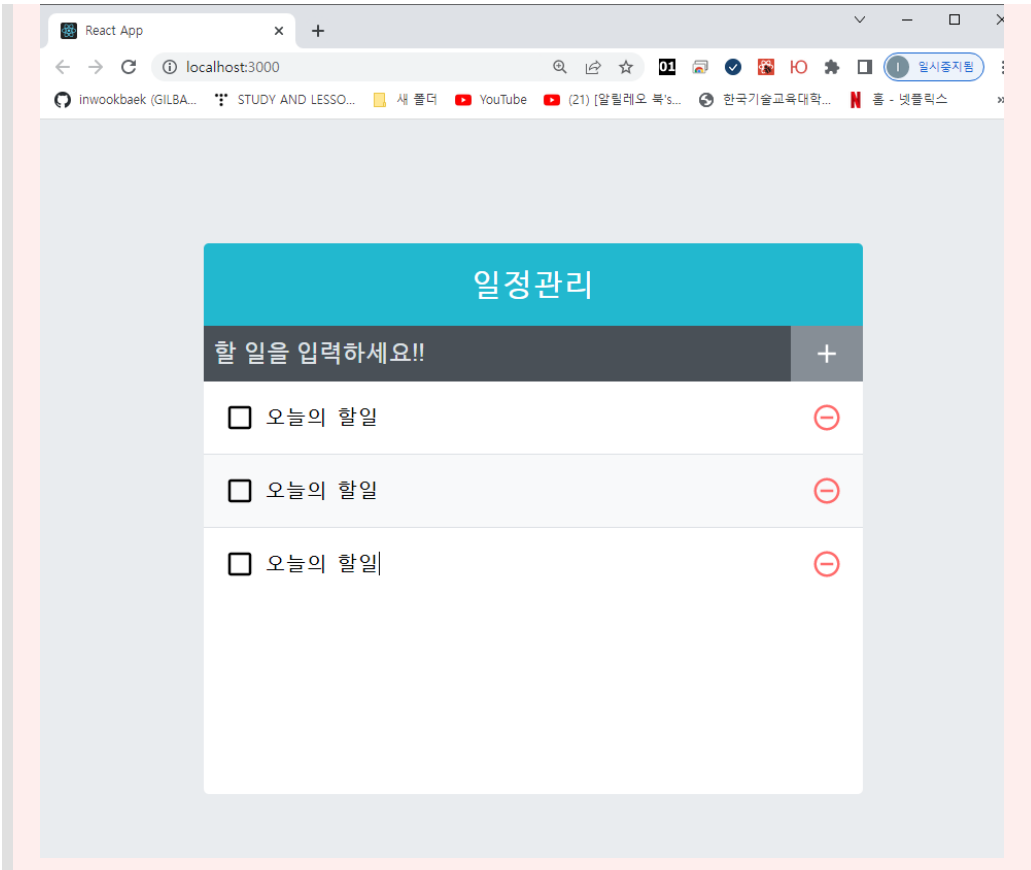
TodoList.scss

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

TodoListItem.scss

```
.TodoListItem {
  padding: 1rem;
  display: flex;
  align-items: center; // 세로 중앙 정렬
  &:nth-child(even) {
    background: #f8f9fa;
  }
  .checkbox {
    cursor: pointer;
    flex: 1; // 차지할 수 있는 영역 모두 차지
    display: flex;
    align-items: center; // 세로 중앙 정렬
    svg {
      // 아이콘
```

```
      font-size: 1.5rem;
    }
    .text {
      margin-left: 0.5rem;
      flex: 1; // 차지할 수 있는 영역 모두 차지
    }
    // 체크되었을 때 보여줄 스타일
    &.checked {
      svg {
        color: #22b8cf;
      }
      .text {
        color: #adb5bd;
        text-decoration: line-through;
      }
    }
  }
  .remove {
    display: flex;
    align-items: center;
    font-size: 1.5rem;
    color: #ff6b6b;
    cursor: pointer;
    &:hover {
      color: #ff8787;
    }
  }

  // 엘리먼트 사이사이에 테두리를 넣어줌
  & + & {
    border-top: 1px solid #dee2e6;
  }
}
```



10.3 기능 구현하기

10.3.1 App에서 todos 상태 사용하기

App.js

- 상태관리를 하는 todos 배열을 TodoList에 props로 전달하고 TodoList에서 TodoItem으로 변환하여 랜더링하기

```
import { useState } from "react";
import TodoInsert from "./components/TodoInsert";
import TodoList from "./components/TodoList";
import TodoTemplate from "./components/TodoTemplate";

const App = () => {

  const [todos, setTodos] = useState([
    {
      id: 1,
      text: 'React.js Study',
      checked: true
    },
    {
      id: 2,
      text: 'Node.js Study',
      checked: true
    },
    {
      id: 3,
      text: 'Todo List App development...',
```

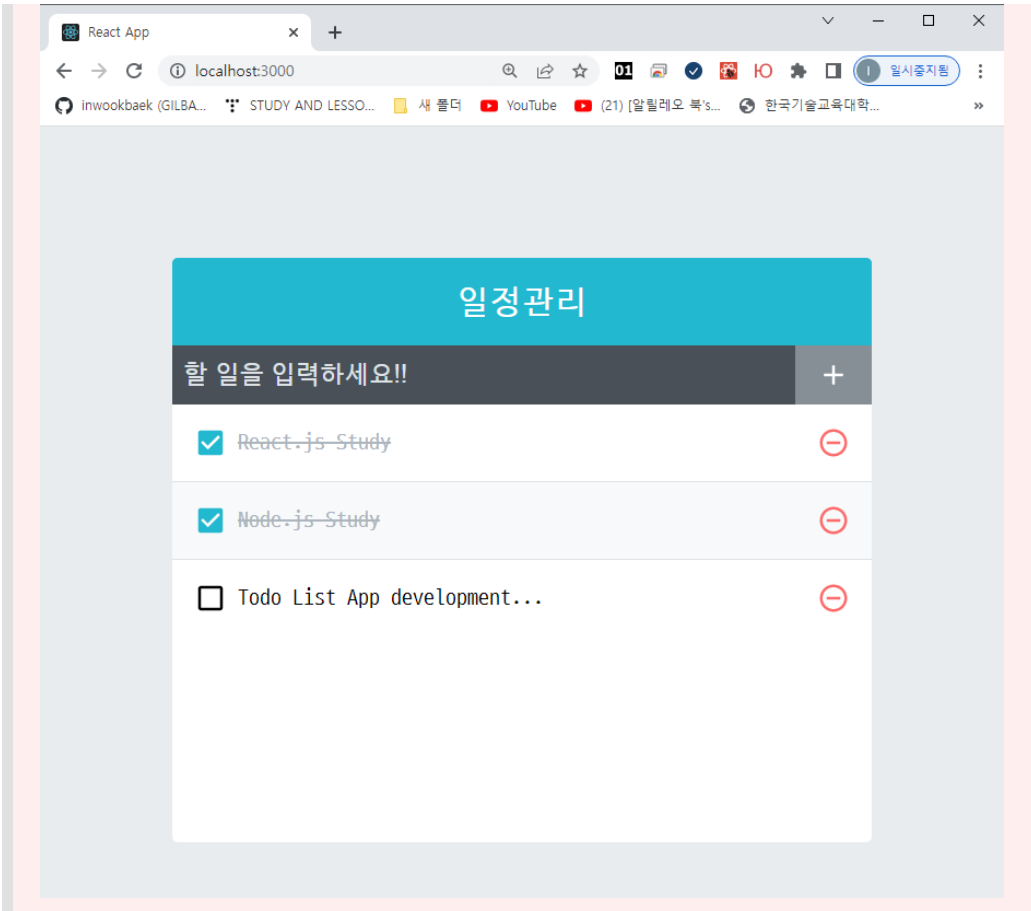
```
      checked: false
    },
  ])
  return (
    <TodoTemplate>
      <TodoInsert />
      <TodoList todos={todos}/>
    </TodoTemplate>
  );
};

export default App;
```

TodoListItem.js

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

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

export default TodoList;
```

TodoListItem.js

```
import {
  MdCheckBoxOutlineBlank,
  MdCheckBox,
  MdRemoveCircleOutline,
} from 'react-icons/md';
import cn from 'classnames'
import './TodoListItem.scss';

const TodoListItem = ({ todo }) => {

  const {text, checked} = todo;

  return (
    <div className="TodoListItem">
      <div className={cn('checkbox', {checked})}>
        {checked ? <MdCheckBox />  : <MdCheckBoxOutlineBlank /> }
        <div className="text">{ text }</div>
      </div>
      <div className="remove">
        <MdRemoveCircleOutline />
      </div>
    </div>
  );
};

export default TodoListItem;
```



10.3.2 항목 추가하기

10.3.3.2.1 TodoInsert value 상태관리하기

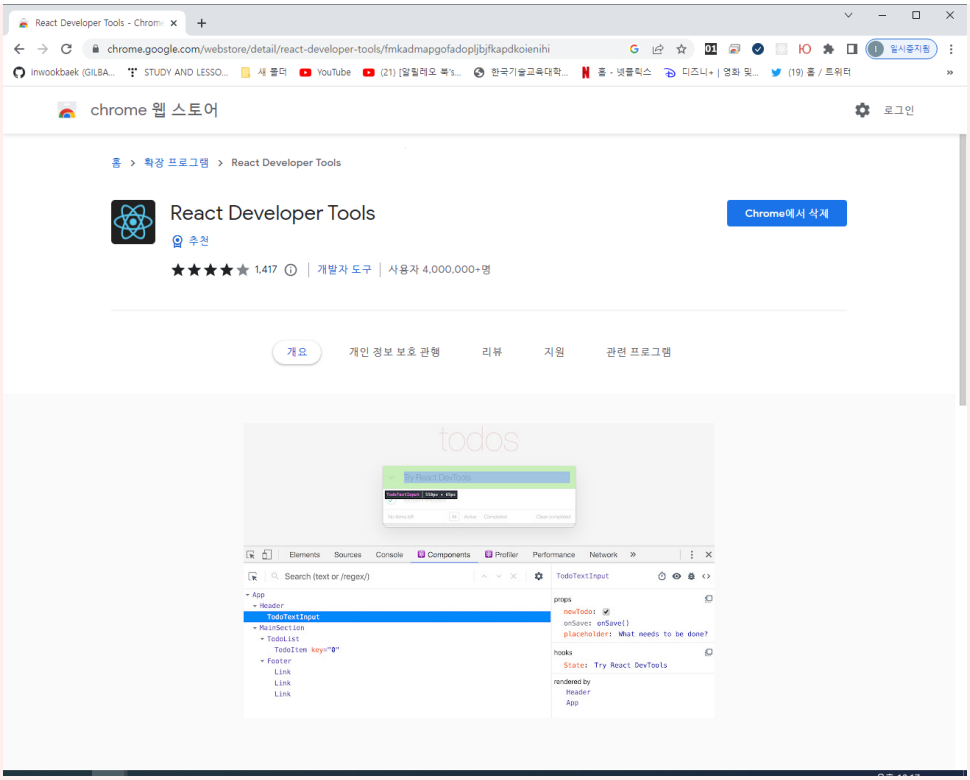
TodoInsert.js

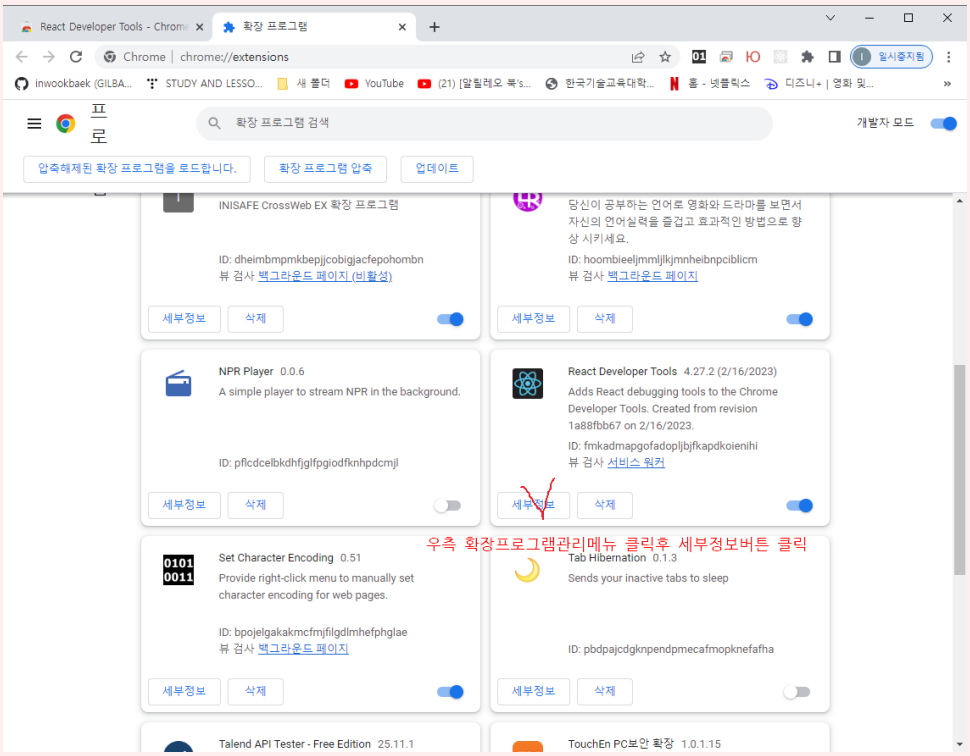
- `useCallback hook을 사용`하여 랜터링될 때 마다 함수를 새로 만드는 것이 아니라 `한번 함수가 생성된 뒤에 재사용`

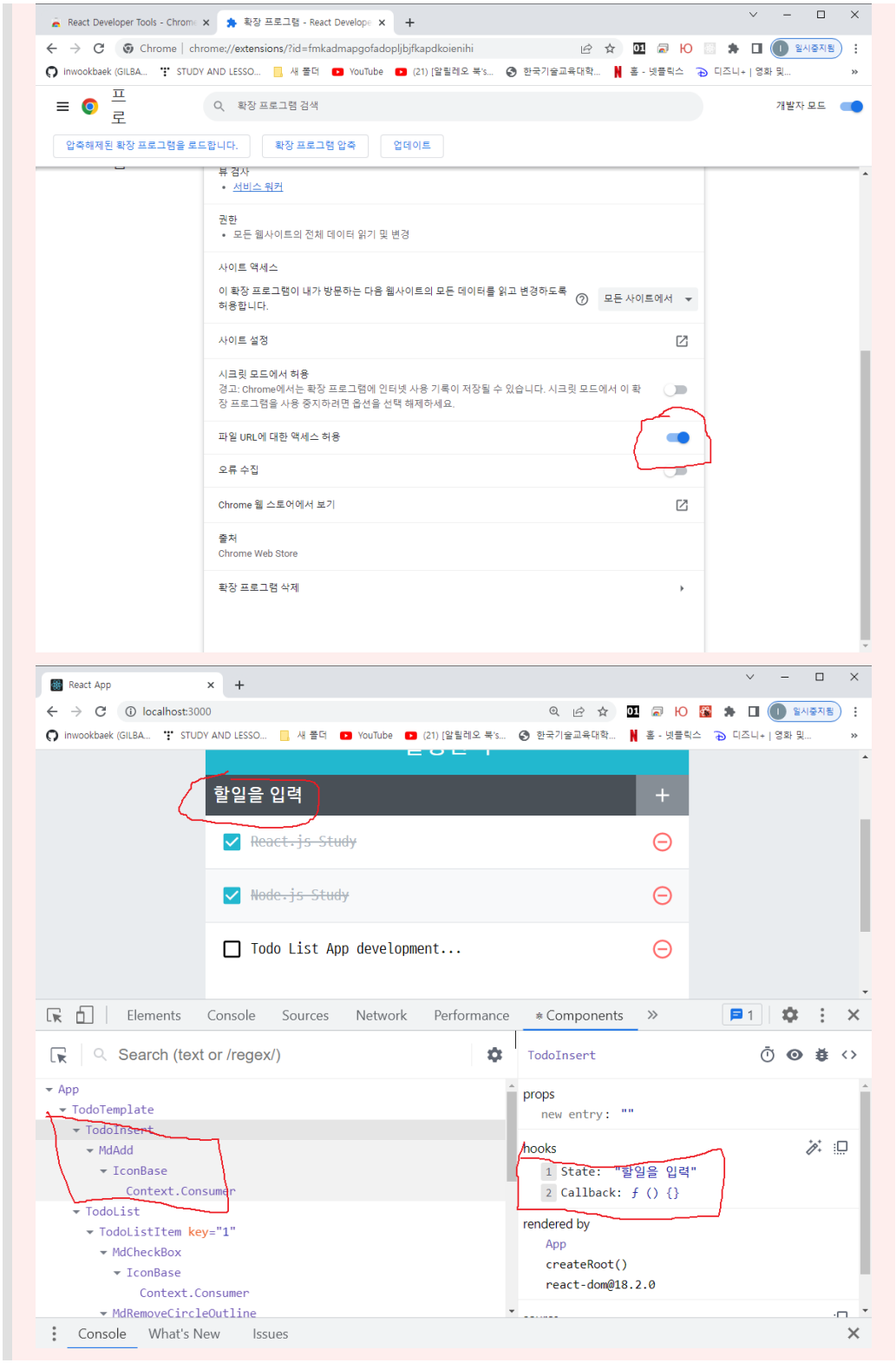
```
import { useState, useCallback } from 'react';
import { MdAdd } from 'react-icons/md';
import './TodoInsert.scss';

const TodoInsert = () => {

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

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

  return (
    <form className='TodoInsert'>
      <input placeholder='할 일을 입력하세요!!'
        value={value}
        onChange={onChange}/>
      <button type='button'>
        <MdAdd />
      </button>
    </form>
  )
```

```
}

export default TodoInsert
```

- input value와 onChange를 설정하지 않더라도 입력할 수 있다.
- 설정하지 않을 경우 리액트컴퍼넌트에서 input에 어떤 값이 입력되어 있는지 추적을 하지 않느다.
- onChange함수 안에서 consloe.log를 찍어 보지 않아도 `업데이트여부를 확인하려면 크롬에 릭액트개발자도구 설치`
- React Developer Tools은 리액트 컴퍼넌트를 심층 분석할 수 있도록 리액트개발팀이 만들었다.
  - 크롬웹스터어에서 React Developer Tools로 검색하여 설치할 수 있다.
  - https://chrome.google.com/webstore/category/extentions


chrome 웹 스토어
로그인
홈 > 확장 프로그램 > React Developer Tools
React Developer Tools
추천
Chrome에서 삭제
★★★★★ 1,417 | 개발자 도구 | 사용자 4,000,000+명
개요
개인 정보 보호 관행
리뷰
지원
관련 프로그램
todos


확장 프로그램
chrome://extensions
확장 프로그램 검색
개발자 모드
압축해제된 확장 프로그램을 로드합니다.
확장 프로그램 압축
업데이트
INISAFE CrossWeb EX 확장 프로그램
ID: dheimbmpmkbepjjcobigjacfepohombn
뷰 검사 백그라운드 페이지 (비활성)
세부정보
삭제
당신이 공부하는 언어로 영화와 드라마를 보면서 자신의 언어실력을 즐겁고 효과적인 방법으로 향상 시키세요.
ID: hoombieeljmmljlkjmnheibnpciblicm
뷰 검사 백그라운드 페이지
NPR Player 0.0.6
A simple player to stream NPR in the background.
ID: pflcdcelbkdhfjglfpgiodfknhpdcmjl
React Developer Tools 4.27.2 (2/16/2023)
Adds React debugging tools to the Chrome Developer Tools. Created from revision 1a88fbb67 on 2/16/2023.
ID: fmkadmapgofadopljbjfkapdkoienihi
뷰 검사 서비스 워커
우측 확장프로그램관리메뉴 클릭후 세부정보버튼 클릭
Set Character Encoding 0.51
Provide right-click menu to manually set character encoding for web pages.
ID: bpojelgakakmcfmjfilgdlmhefphglae
뷰 검사 백그라운드 페이지
Tab Hibernation 0.1.3
Sends your inactive tabs to sleep
ID: pbdpajcdgknpendpmecafmopknefafha
Talend API Tester - Free Edition 25.11.1
TouchEn PC보안 확장 1.0.1.15



10.3.2.3 todos 배열에 새 객체 추가

- App컴퍼넌트에 todos 배열에 새 객체를 추가하는 onInsert함수 작성
- 추가할 때 `id값은 useRef로 관리` 하는 것은 랜처링되는 정보가 아니라 useState르 사용할 필요가 없다.
- onInsert함수는 컴퍼넌트의 성능을 위해서 `useCallback hook으로 감싸서 선언`
- props로 전달해야 할 함수를 만들 때는 useCallback을 사용하여 함수를 감싸는 것을 습관화할 것

App.js

```
import { useCallback, useRef, useState } from "react";
import TodoTemplate from "./components/TodoTemplate";
```

```
import TodoInsert from "./components/TodoInsert";
import TodoList from "./components/TodoList";

const App = () => {

  const [todos, setTodos] = useState([
    {
      id: 1,
      text: 'React.js Study',
      checked: true
    },
    {
      id: 2,
      text: 'Node.js Study',
      checked: true
    },
    {
      id: 3,
      text: 'Todo List App development...',
      checked: false
    },
  ]);

  // 고유값으로 사용할 id를 ref를 사용하여 변수에 저장
  const nextId = useRef(4);

  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} />
      <TodoList todos={todos}/>
    </TodoTemplate>
  );
};

export default App;
```

10.3.2.4 TodoInsert onSubmit 이벤트 설정

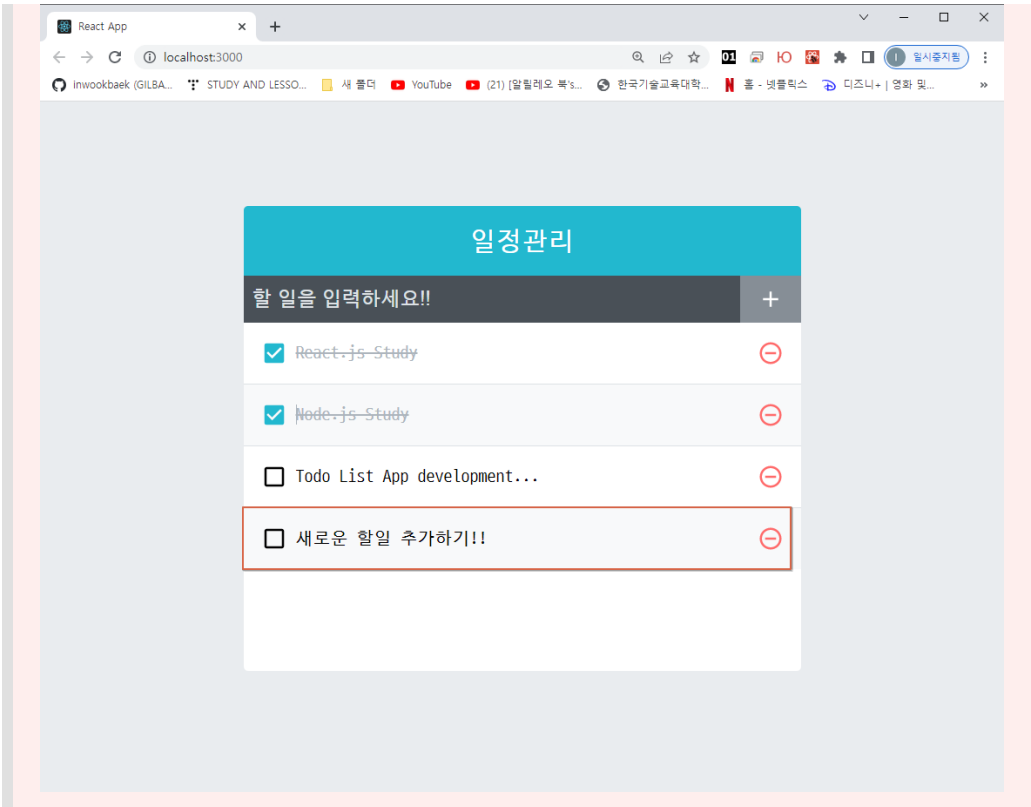
TodoInsert.js

```
import { useState, useCallback } from 'react';
import { MdAdd } from 'react-icons/md';
import './TodoInsert.scss';

const TodoInsert = ({ onInsert }) => {
```

```
  const [value, setValue] = useState('');

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

  const onSubmit = useCallback(
    e => {
      onInsert(value);
      setValue('');

      // submit 이벤트는 브라우저에서 새로고침을 발생시킵니다.
      // 이를 방지하기 위하여 이 함수를 호출합니다.
      e.preventDefault();
    }, [onInsert, value]
  );

  return (
    <form className='TodoInsert' onSubmit={onSubmit}>
      <input placeholder='할 일을 입력하세요!!'
        value={value}
        onChange={onChange}/>
      <button type='submit'>
        <MdAdd />
      </button>
    </form>
  )
}
export default TodoInsert
```

- `obSubmit함수와 form에 onSubmit로 설정` 이 함수가 호출되면 props로 받아 온 onInsert함수에 현재 value값을 파라미터로 전달
- onSubmit 이벤트는 브라우저를 새로고침을 한다. 이를 방지하기 위해 e.preventDefault()함수를 호출
- onSubmit대사 onClick으로 처리할 수 있다.

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

    ... 생략 ...

      <button type='button' onClick={onClick}>
        <MdAdd />
      </button>
```



10.3.3 삭제하기

10.3.3.1 todos 배열에서 id로 삭제하기

App.js

```
import { useCallback, useRef, useState } from "react";
import TodoTemplate from "./components/TodoTemplate";
import TodoInsert from "./components/TodoInsert";
import TodoList from "./components/TodoList";

const App = () => {

  const [todos, setTodos] = useState([
    {
      id: 1,
      text: 'React.js Study',
      checked: true
    },
    {
      id: 2,
      text: 'Node.js Study',
      checked: true
    },
    {
      id: 3,
      text: 'Todo List App development...',
      checked: false
    },
  ]);

  // 고유값으로 사용할 id를 ref를 사용하여 변수에 저장
  const nextId = useRef(4);
```

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

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

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

export default App;
```

10.3.3.2 TodoListItem에서 삭제함수 호출

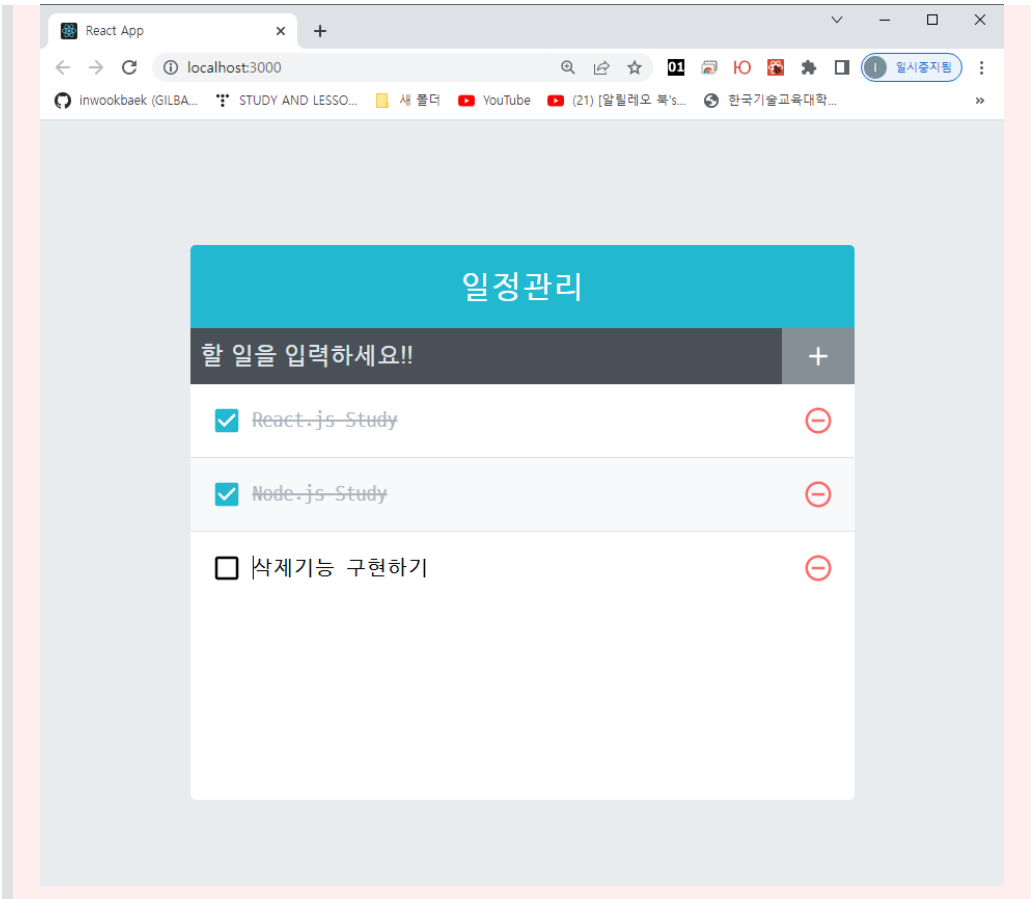
TodoList.js

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

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

export default TodoList;
```

TodoListItem.js

```
import {
  MdCheckBoxOutlineBlank,
  MdCheckBox,
  MdRemoveCircleOutline,
} from 'react-icons/md';
import cn from 'classnames'
import './TodoListItem.scss';
```

```
const TodoListItem = ({ todo, onRemove }) => {

  const {id, text, checked} = todo;

  return (
    <div className="TodoListItem">
      <div className={cn('checkbox', {checked})}>
        {checked ? <MdCheckBox />  : <MdCheckBoxOutlineBlank /> }
        <div className="text">{ text }</div>
      </div>
      <div className="remove" onClick={() => onRemove(id)}>
        <MdRemoveCircleOutline />
      </div>
    </div>
  );
};

export default TodoListItem;
```



10.3.4 수정하기

App.js

```
import { useCallback, useRef, useState } from "react";
import TodoTemplate from "./components/TodoTemplate";
import TodoInsert from "./components/TodoInsert";
import TodoList from "./components/TodoList";

const App = () => {

  const [todos, setTodos] = useState([
    {
```

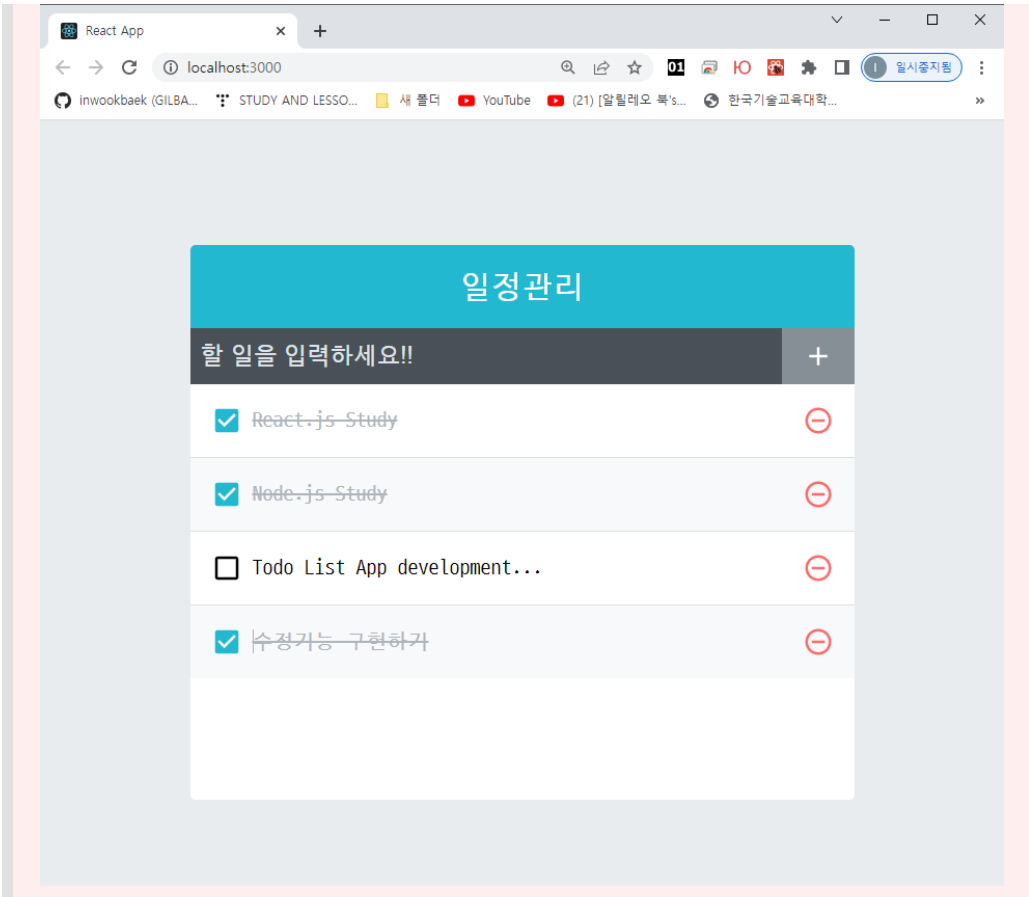
```
      id: 1,
      text: 'React.js Study',
      checked: true
    },
    {
      id: 2,
      text: 'Node.js Study',
      checked: true
    },
    {
      id: 3,
      text: 'Todo List App development...',
      checked: false
    },
  ]);

  // 고유값으로 사용할 id를 ref를 사용하여 변수에 저장
  const nextId = useRef(4);

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

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

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

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

export default App;
```

TodoList.js

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

const TodoList = ({ todos, onRemove, onToggle }) => {
  return (
    <div className="TodoList">
      {todos.map(todo => (
        <TodoListItem todo={todo} key={todo.id} onRemove={onRemove} onToggle=
{onToggle}/>
      ))}
    </div>
  );
};

export default TodoList;
```

TodoListItem.js

```
import {
  MdCheckBoxOutlineBlank,
  MdCheckBox,
  MdRemoveCircleOutline,
} from 'react-icons/md';
import cn from 'classnames'
import './TodoListItem.scss';

const TodoListItem = ({ todo, onRemove, onToggle }) => {

  const {id, text, checked} = todo;

  return (
    <div className="TodoListItem">
      <div className={cn('checkbox', {checked})} onClick={() => onToggle(id)}>
        {checked ? <MdCheckBox />  : <MdCheckBoxOutlineBlank /> }
        <div className="text">{ text }</div>
      </div>
      <div className="remove" onClick={() => onRemove(id)}>
        <MdRemoveCircleOutline />
      </div>
    </div>
  );
};

export default TodoListItem;
```


React App
localhost:3000
일정관리
할 일을 입력하세요!!
React.js Study
Node.js Study
Todo List App development...
수정기능 구현하기