

일정 관리 웹 애플리케이션 만들기

```
$ yarn create react-app todo-app
```

프로젝트가 생성되면 todo-app 디렉터리로 들어가서 필요한 라이브러리를 설치 함

```
$ cd todo-app
```

```
$ yarn add sass classnames react-icons
```

sass : 자주 사용되는 CSS 전처리기 중 하나로 확장된 CSS 문법을 사용하여 CSS 코드를 더욱 쉽게 작성해 줌

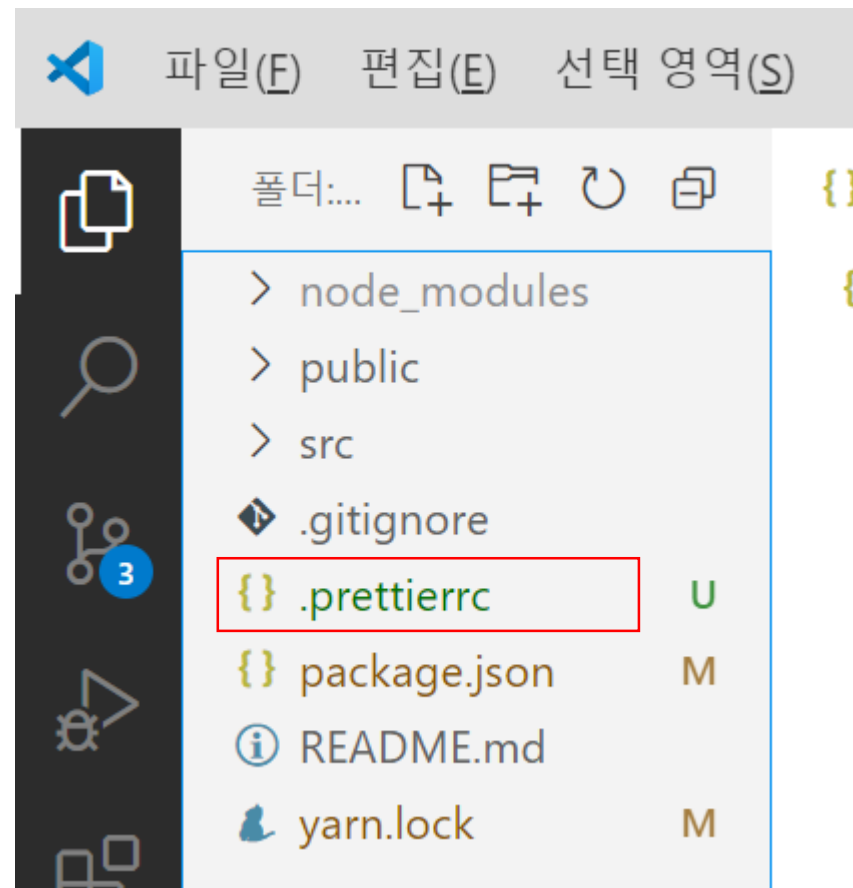
classnames : 조건부 스타일링을 좀 더 편하게 하기 위해 설치

react-icons : 리액트에서 다양하고 예쁜 아이콘을 사용할 수 있는 라이브러리

Prettier 설정

코드 스타일을 깔끔하게 정리하기 위하여 최상위 디렉터리에 .prettierrc 파일을 생성

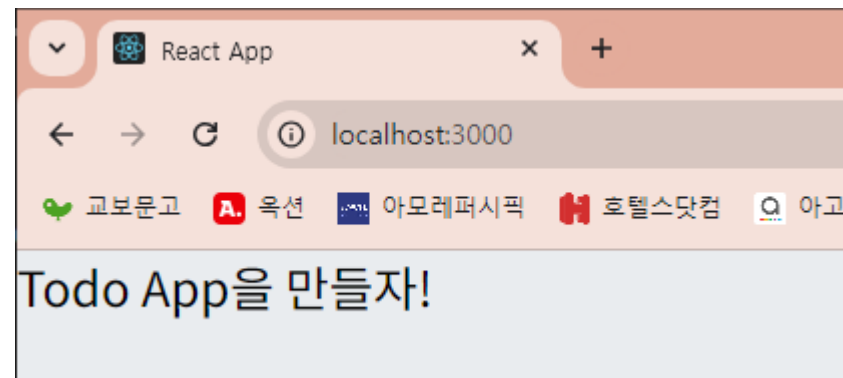
```
1 {  
2   "singleQuote": true,  
3   "semi": true,  
4   "useTabs": false,  
5   "tabWidth": 2,  
6   "trailingComma": "all",  
7   "printWidth": 80  
8 }  
9
```



index.css 수정

프로젝트의 글로벌 스타일 파일이 들어 있는 index.css 파일의 내용을 전부 삭제하고 다음과 같이 수정

```
1 body {  
2   margin: 0;  
3   padding: 0;  
4   background: #e9ecef;  
5 }
```



App 컴포넌트 초기화

```
1 const App = () => {  
2   return <div> Todo App을 만들자! </div>  
3 };  
4  
5 export default App;
```

UI 구성하기

1. `TodoTemplate` : 화면을 가운데에 정렬시켜 주며, 앱 타이틀(일정관리)을 보여줌
children 으로 내부 JSX를 props로 받아 와서 랜더링 해 줌
2. `TodoInsert` : 새로운 항목을 입력하고 추가할 수 있는 컴포넌트임
state를 통해 인풋의 상태를 관리 함
3. `TodoListItem` : 각 할 일 항목에 대한 정보를 보여 주는 컴포넌트
todo 객체를 props로 받아 와서 상태에 따른 다른 스타일의 UI를 보여줌
4. `TodoList` : todo 배열을 props로 받아 온 후, 이를 배열 내장 함수 map을 사용해서 여러 개의 `TodoListItem` 컴포넌트로 변환하여 보여줌

이렇게 총 네 개의 컴포넌트들은 src 디렉터리에 components 라는 디렉터리를 생성하여 그 안에 저장.

TodoTemplate 만들기 (TodoTemplate.scss)

```
1 .TodoTemplate {  
2   width: 512px;  
3   // width가 주어진 상태에서 좌우 중앙 정렬  
4   margin-left: auto;  
5   margin-right: auto;  
6   margin-top: 6rem;  
7   border-radius: 4px;  
8   overflow: hidden;  
9  
10  .app-title {  
11    background: #22b8cf;  
12    color: white;  
13    height: 4rem;  
14    font-size: 1.5rem;  
15    display: flex;  
16    align-items: center;  
17    justify-content: center;  
18  }  
19  .content {  
20    background: white;  
21  }  
22 }
```

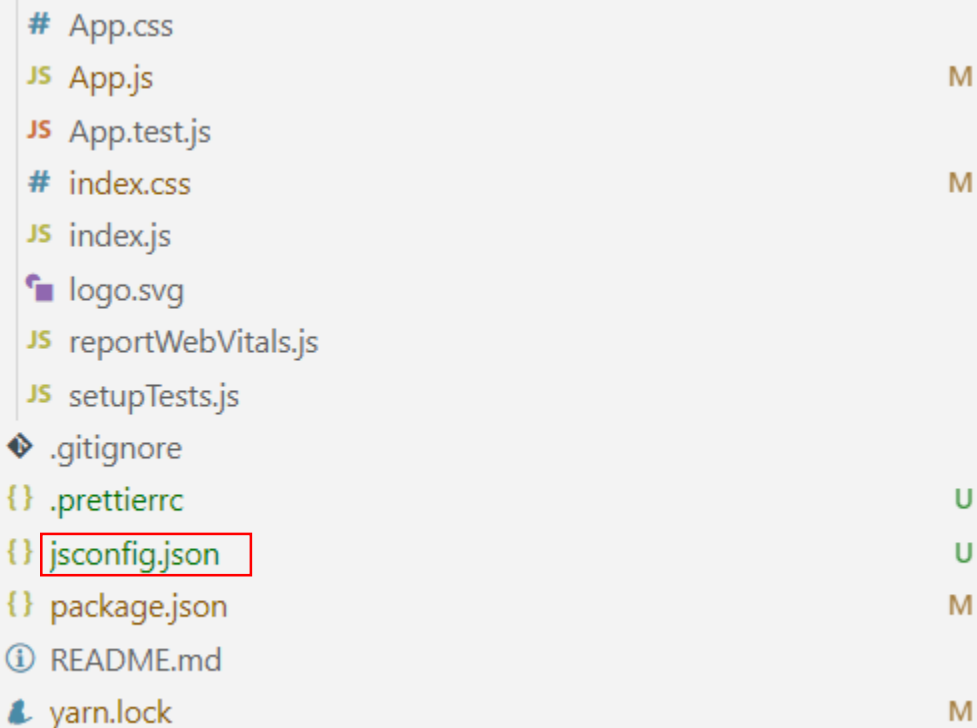
TodoTemplate 만들기 (TodoTemplate.js)

```
1 import React from 'react';
2 import './TodoTemplate.scss';
3
4 const TodoTemplate = ({ children }) => {
5   return (
6     <div className="TodoTemplate">
7       <div className="app-title">일정 관리</div>
8       <div className="content">{children}</div>
9     </div>
10  );
11 };
12
13 export default TodoTemplate;
```

VS Code 에디터에서 import 를 넣지 않고 바로 컴포넌트를 사용하려고 하면, VS Code 에디터에서 자동 완성 기능이 나타남

그러나 TodoTemplate.js 컴포넌트가 VS Code에서 다른 탭으로 열려 있지 않으면 자동 완성이 작동하지 않으므로, 닫혀 있는 파일에도 자동 완성이 제대로 작동하려면 프로젝트 최 상단 디렉터리에 jsconfig.json 파일을 만들어 줌

```
1 {  
2   "compilerOptions": {  
3     "target": "es6"  
4   }  
5 }
```



File	Status
# App.css	
JS App.js	M
JS App.test.js	
# index.css	M
JS index.js	
logo.svg	
JS reportWebVitals.js	
JS setupTests.js	
.gitignore	
.prettierrc	U
jsconfig.json	U
package.json	M
README.md	
yarn.lock	M



TodoInsert 만들기 (TodoInsert.scss 작성)

```
1 .TodoInsert {  
2   display: flex;  
3   background: #495057;  
4   input {  
5     // 기본 스타일 초기화  
6     background: none;  
7     outline: none;  
8     border: none;  
9     padding: 0.5rem;  
10    font-size: 1.125rem;  
11    line-height: 1.5;  
12    color: white;  
13    &::placeholder {  
14      color: #dee2e6;  
15    }  
16    // 버튼을 제외한 영역을 모두 차지하기  
17    flex: 1;  
18  }
```

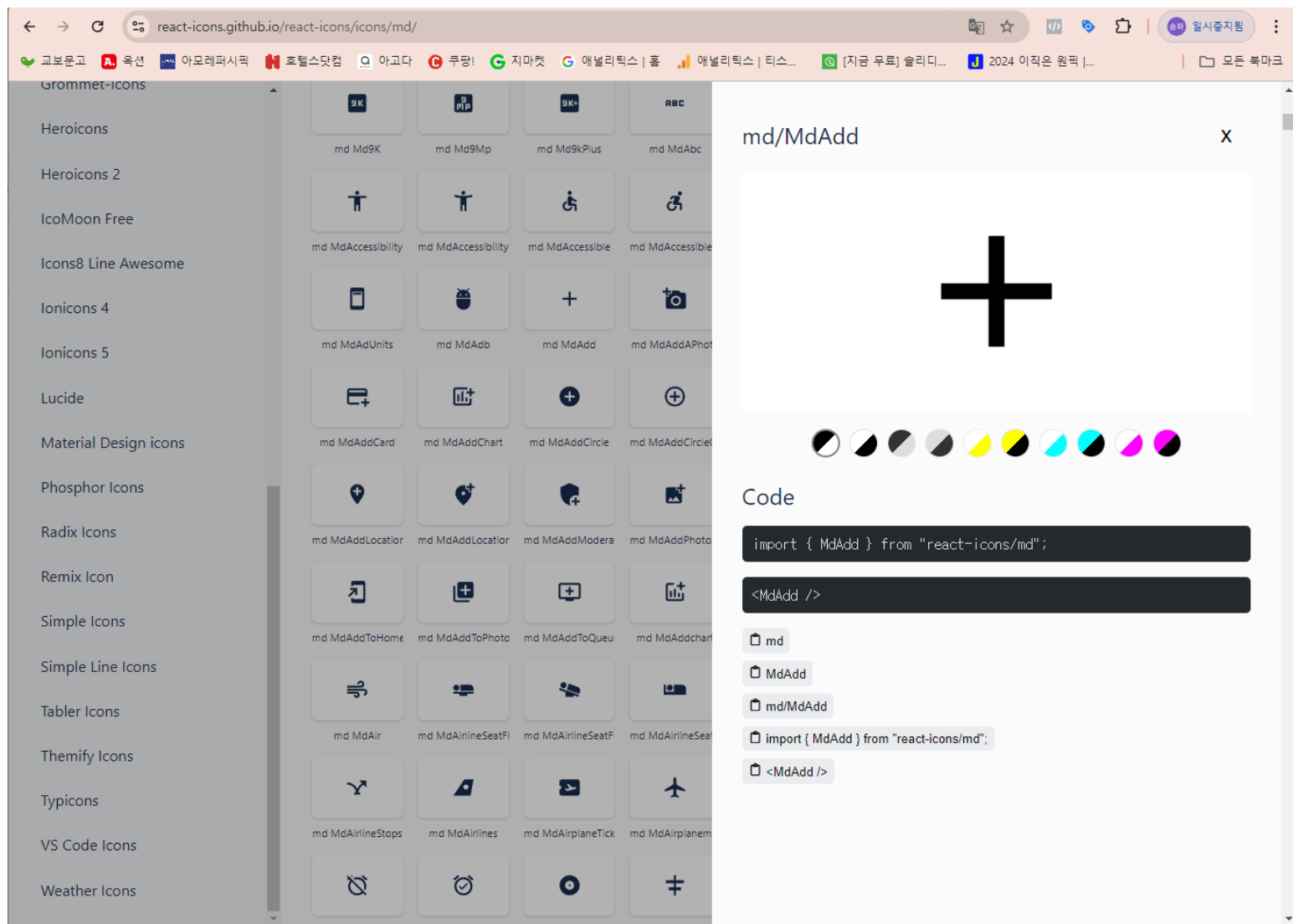
```
19 button {  
20   // 기본 스타일 초기화  
21   background: none;  
22   outline: none;  
23   border: none;  
24   background: #868e96;  
25   color: white;  
26   padding-left: 1rem;  
27   padding-right: 1rem;  
28   font-size: 1.5rem;  
29   display: flex;  
30   align-items: center;  
31   cursor: pointer;  
32   transition: 0.1s background ease-in;  
33   &:hover {  
34     background: #adb5bd;  
35   }  
36 }  
37 }
```

TodoInsert 만들기 (TodoInsert.js 작성)

```
1 import { MdAdd } from 'react-icons/md';
2 import './TodoInsert.scss';
3
4 const TodoInsert = () => {
5   return (
6     <form className="TodoInsert">
7       <input placeholder="할 일을 입력하세요" />
8       <button type="submit">
9         <MdAdd />
10      </button>
11    </form>
12  );
13 };
14
15 export default TodoInsert;
```

react-icons 의 사용

<https://react-icons.github.io/react-icons/> 사이트에 수많은 아이콘이 제공됨



App.js 에서 렌더링

```
1 import TodoTemplate from "../components/TodoTemplate";
2 import TodoInsert from "../components/TodoInsert";
3
4 const App = () => {
5   return (
6     <TodoTemplate>
7     <TodoInsert />
8     </TodoTemplate>
9   )
10 };
11
12 export default App;
```

일정 관리

할 일을 입력하세요

+

TodoListItem.scss 작성

```
1 .TodoListItem {
2   padding: 1rem;
3   display: flex;
4   align-items: center; // 세로 중앙 정렬
5   &:nth-child(even) {
6     background: #f8f9fa;
7   }
8   .checkbox {
9     cursor: pointer;
10    flex: 1; // 차지할 수 있는 영역 모두 차지
11    display: flex;
12    align-items: center; // 세로 중앙 정렬
13    svg {
14      // 아이콘
15      font-size: 1.5rem;
16    }
17    .text {
18      margin-left: 0.5rem;
19      flex: 1; // 차지할 수 있는 영역 모두 차지
20    }
21    // 체크되었을 때 보여줄 스타일
22    &.checked {
23      svg {
24        color: #22b8cf;
25      }
26    }
27  }
28 }
```

```
26 .text {
27   color: #adb5bd;
28   text-decoration: line-through;
29 }
30 }
31 }
32 .remove {
33   display: flex;
34   align-items: center;
35   font-size: 1.5rem;
36   color: #ff6b6b;
37   cursor: pointer;
38   &:hover {
39     color: #ff8787;
40   }
41 }
42 // 엘리먼트 사이사이에 테두리를 넣어줌
43 & + & {
44   border-top: 1px solid #dee2e6;
45 }
46 }
47 }
```

이제 일정 관리 항목이 보일
TodoListItem 과 TodoList를 작성

TodoListItem.js 작성

```
1 import React from 'react';
2 import {
3   MdCheckBoxOutlineBlank,
4   MdCheckBox,
5   MdRemoveCircleOutline,
6 } from 'react-icons/md';
7 import './TodoListItem.scss';
8
9 const TodoListItem = () => {
10   return (
11     <div className="TodoListItem">
12       <div className="checkbox">
13         <MdCheckBoxOutlineBlank />
14         <div className="text"> 할 일 </div>
15       </div>
16       <div className="remove">
17         <MdRemoveCircleOutline />
18       </div>
19     </div>
20   );
21 };
22
23 export default TodoListItem;
```

TodoList.scss 작성

```
1 .TodoList {  
2   min-height: 320px;  
3   max-height: 513px;  
4   overflow-y: auto;  
5 }
```

TodoList.js 작성

```
1 import React from 'react';  
2 import TodoListItem from './TodoListItem';  
3 import './TodoList.scss';  
4  
5 const TodoList = () => {  
6   return (  
7     <div className="TodoList">  
8       <TodoListItem />  
9       <TodoListItem />  
10      <TodoListItem />  
11    </div>  
12  );  
13 };  
14  
15 export default TodoList;
```


App.js 랜더링

```
1 import TodoTemplate from "../components/TodoTemplate";
2 import TodoInsert from "../components/TodoInsert";
3 import TodoList from "../components/TodoList";
4
5 const App = () => {
6   return (
7     <TodoTemplate>
8       <TodoInsert />
9       <TodoList />
10    </TodoTemplate>
11  );
12 };
13
14 export default App;
```



App 에서 useState를 사용하여 todos 라는 상태를 정의하고, todos를 TodoList의 props로 전달

```
1 import { useState } from "react";
2 import TodoTemplate from "../components/TodoTemplate";
3 import TodoInsert from "../components/TodoInsert";
4 import TodoList from "../components/TodoList";
5
6 const App = () => {
7   const [todos, setTodos] = useState([
8     {
9       id: 1,
10      text: '리액트 기초 알아보기',
11      checked: true,
12    },
13    {
14      id: 2,
15      text: '컴포넌트 스타일링 해보기',
16      checked: true,
17    },
```

```
18    {
19      id: 3,
20      text: '일정 관리 앱 만들어 보기',
21      checked: false,
22    }
23  ])
24  return (
25    <TodoTemplate>
26      <TodoInsert />
27      <TodoList todos={todos} />
28    </TodoTemplate>
29  )
30 };
31
32 export default App;
```

- todos 배열 안에 들어 있는 객체에는 각 항목의 고유 id, 내용, 완료 여부를 알려주는 값이 포함되어 있음
- 이 배열은 TodoList에 props로 전달되는데, TodoList에서 이 값을 받아 온 후 TodoItem으로 변환하여 렌더링 하도록 설정

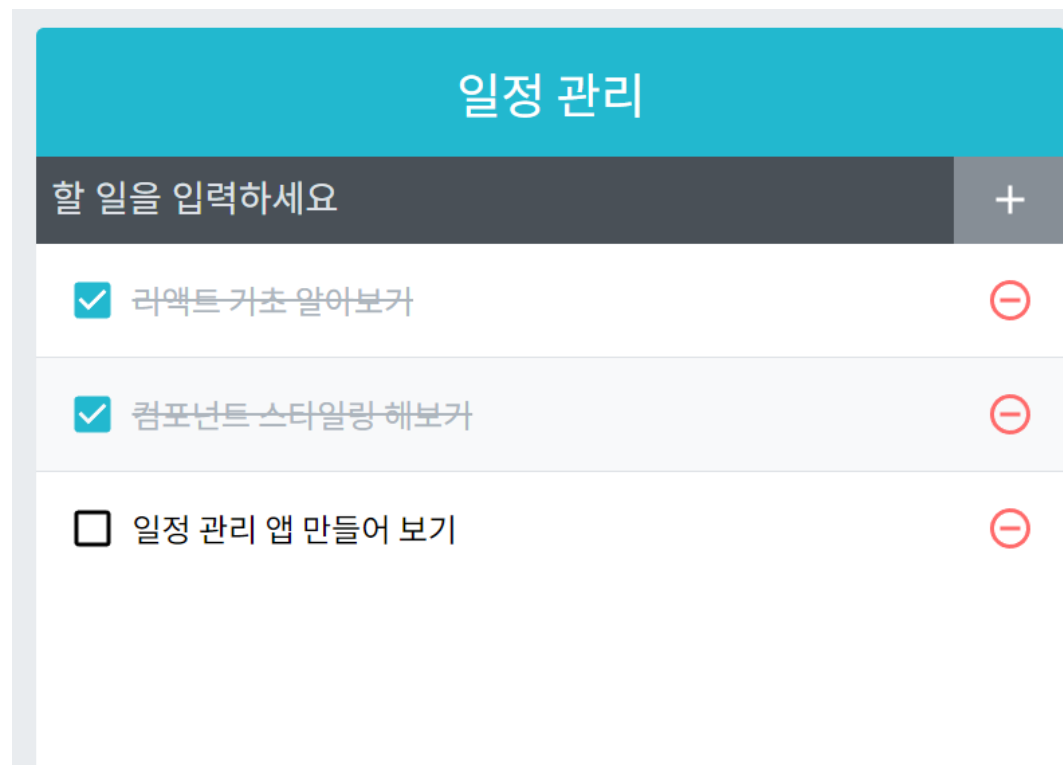
TodoList.js

```
1 import React from 'react';
2 import TodoListItem from './TodoListItem';
3 import './TodoList.scss';
4
5 const TodoList = ({ todos }) => {
6   return (
7     <div className="TodoList">
8       {todos.map(todo => (
9         <TodoListItem todo={todo} key={todo.id} />
10       ))}
11     </div>
12   );
13 };
14
15 export default TodoList;
```

- props로 받아 온 todos 배열을 배열 내장 함수 map을 통해 TodoListItem 으로 이루어진 배열로 변환하여 랜더링 해 줌
- map을 사용하여 컴포넌트로 변환할 때는 key props로 전달해 주어야 한다고 했는데, 여기서 사용되는 key 값은 각 항목마다 가지고 있는 고유값인 id를 넣어주고, todo 데이터는 통째로 props로 전달해 줌
- 여러 종류의 값을 전달해야 하는 경우에는 객체로 통째로 전달하는 편이 나중에 성능 최적화를 할 때 편리함

이제 TodoListItem 컴포넌트에서 받아 온 todo 값에 따라 제대로 된 UI를 보여줄 수 있도록 컴포넌트를 수정
조건부 스타일링을 위해 classNames를 사용함

```
1 import React from 'react';
2 import {
3   MdCheckBoxOutlineBlank,
4   MdCheckBox,
5   MdRemoveCircleOutline,
6 } from 'react-icons/md';
7 import './TodoListItem.scss';
8 import cn from 'classnames';
9
10
11 const TodoListItem = ({ todo }) => {
12   const { text, checked } = todo;
13   return (
14     <div className="TodoListItem">
15       <div className={cn('checkbox', { checked })}>
16         {checked ? <MdCheckBox /> : <MdCheckBoxOutlineBlank /> }
17         <div className="text">{text}</div>
18       </div>
19       <div className="remove">
20         <MdRemoveCircleOutline />
21       </div>
22     </div>
23   );
24 };
25
26 export default TodoListItem;
```



항목 추가 기능 구현

- TodoInsert 컴포넌트에서 인풋에 입력하는 값을 관리할 수 있도록 useState를 사용하여 value라는 상태를 정의함
- 추가로 인풋에 넣어 줄 onChange 함수도 작성해 줌
- 이 과정에서 컴포넌트가 리렌더링될 때마다 함수를 새로 만드는 것이 아니라, 한 번 함수를 만들고 재사용할 수 있도록 useCallback Hook를 사용함

인풋에 텍스트를 입력하면 정상 작동 함

일정 관리	
이병일	+
<input checked="" type="checkbox"/> 라랙트 기초 알아보기	⊖
<input checked="" type="checkbox"/> 컴포넌트 스타일링 해보기	⊖
<input type="checkbox"/> 일정 관리 앱 만들어 보기	⊖

```
1 import { useState, useCallback } from 'react';
2 import { MdAdd } from 'react-icons/md';
3 import './TodoInsert.scss';
4
5 const TodoInsert = () => {
6   const [value, setValue] = useState("");
7   const onChange = useCallback(e=>{
8     setValue(e.target.value);
9   },[]);
10  return (
11    <form className="TodoInsert">
12      <input
13        placeholder="할 일을 입력하세요"
14        value={value}
15        onChange={onChange}
16      />
17      <button type="submit">
18        <MdAdd />
19      </button>
20    </form>
21  );
22 };
23 export default TodoInsert;
```

todos 배열에 새 객체 추가하기

- 새 객체를 추가하는 onInsert 함수를 작성
- 이 함수에서는 새로운 객체를 만들 때 마다 id 값에 1 씩 더해 주어야 함
- Id 값은 useRef를 사용하여 관리함
- useState가 아닌 useRef를 사용하여 컴포넌트에서 사용할 변수를 만드는 이유는 id 값은 렌더링되는 정보가 아니기 때문에 이 값은 화면에 보이지 않고, 이 값이 바뀐다고 해서 컴포넌트가 리렌더링될 필요도 없음
- 단순히 새로운 항목을 만들 때 참조되는 값일 뿐
- 또한 onInsert 함수는 컴포넌트의 성능을 아낄 수 있도록 useCallback으로 감싸주어야 함
- 보통 props로 전달해야 할 함수를 만들 때는 useCallback을 사용하여 함수를 감싸는 것을 습관화 해야 함

App.js 수정

```
1 import { useState, useRef, useCallback } from "react";
2 import TodoTemplate from "../components/TodoTemplate";
6 const App = () => {
18   {
19     id: 3,
20     text: '일정 관리 앱 만들어 보기',
21     checked: false,
22   }
23 ]
24
25 // 고유키값으로 사용될 id
26 // ref를 사용하여 변수 담기
27
28 const nextId = useRef(4);
29 const onInsert = useCallback(
30   text => {
31     const todo = {
32       id: nextId.current,
33       text,
34       checked: false,
35     };
36     setTodos(todos.concat(todo));
37     nextId.current += 1;
38   },
39   [todos],
40 )
41 return (
42   <TodoTemplate>
43     <TodoInsert onInsert={onInsert} />
44     <TodoList todos={todos} />
45   </TodoTemplate>
46 )
```

TodoInsert에서 onSubmit 이벤트 설정하기

버튼을 클릭하면 발생할 이벤트로 App에서 TodoInsert에 넣어 준 onInsert 함수에 현재 useState를 통해 관리하고 있는 value 값을 파라미터로 넣어서 호출함

```
1 import { useState, useCallback } from 'react';
2 import { MdAdd } from 'react-icons/md';
3 import './TodoInsert.scss';
4
5 const TodoInsert = ({ onInsert }) => {
6   const [value, setValue] = useState("");
7   const onChange = useCallback(e => {
8     setValue(e.target.value);
9   }, []);
10
11   const onSubmit = useCallback(
12     e => {
13       onInsert(value);
14       setValue(""); // value 값 초기화
15       // submit 이벤트는 브라우저에서 새로고침을 발생시킴
16       // 이를 방지하기 위해 이 함수를 호출함
17       e.preventDefault();
18     }, [onInsert, value]
19   );
```

```
20
21   return (
22     <form className="TodoInsert" onSubmit={onSubmit}>
23       <input
24         placeholder="할 일을 입력하세요"
25         value={value}
26         onChange={onChange}
27       />
28       <button type="submit">
29         <MdAdd />
30       </button>
31     </form>
32   );
33 };
34 export default TodoInsert;
35
```


- onSubmit 이라는 함수를 만들고, 이를 form 의 onSubmit으로 설정
- 이 함수가 호출되면 props로 받아 온 onInsert 함수에 현재 value 값을 파라미터로 넣어서 호출하고, 현재 value 값을 초기화 함
- 추가로 onSubmit 이벤트는 브라우저를 새로고침 하는데, 이때 e.preventDefault() 함수를 호출하면 새로고침을 방지할 수 있음

일정 관리

할 일을 입력하세요 +

☒ 라랙트 기초 알아보기

⊖

☒ 컴포넌트 스타일링 해보기

⊖

☐ 일정 관리 앱 만들어 보기

⊖

☐ 새 데이터를 추가해 보자

⊖

지우기 기능 구현하기

- 리액트 컴포넌트에서 배열의 불변성을 지키면서 배열 원소를 제거해야 할 경우, 배열 내장 함수인 `filter`를 사용함
- `filter` 함수는 기본의 배열은 그대로 둔 상태에서 특정 조건을 만족하는 원소들만 따로 추출하여 새로운 배열을 만들어 줌
- App 컴포넌트에 `id`를 파라미터로 받아 와서 `id`를 가진 항목을 `todos` 배열에서 지우는 함수를 작성

```
41 |  
42 | const onRemove = useCallback(  
43 |   id => {  
44 |     setTodos(todos.filter(todo => todo.id !== id));  
45 |   },[todos],  
46 | )  
47 |  
48 | return (  
49 |   <TodoTemplate>  
50 |     <TodoInsert onInsert={onInsert} />  
51 |     <TodoList todos={todos} onRemove = {onRemove} />  
52 |   </TodoTemplate>  
53 | )  
54 | };  
55 |  
56 | export default App;
```

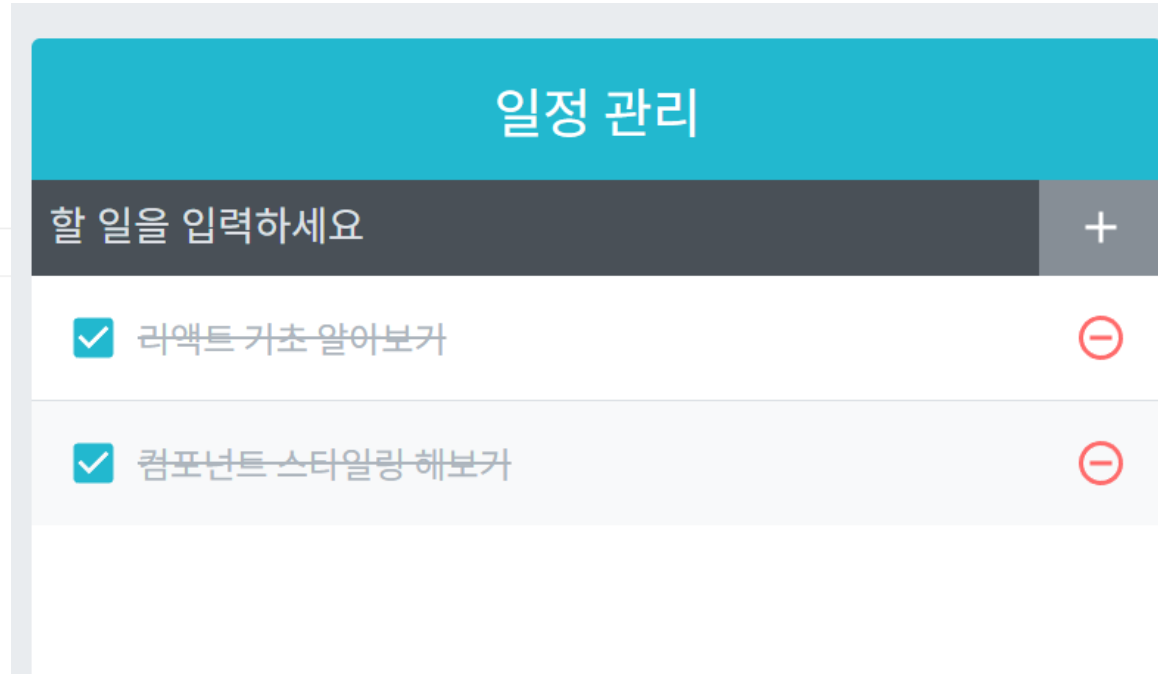
TodoListItem 에서 삭제 함수 호출하기

- TodoListItem에서 방금 만든 onRemove 함수를 사용하려면 우선 TodoList 컴포넌트를 거쳐야 함
- 다음과 같이 props로 받아온 onRemove 함수를 TodoListItem에 그대로 전달해 줌

```
1 import React from 'react';
2 import TodoListItem from './TodoListItem';
3 import './TodoList.scss';
4
5 const TodoList = ({ todos, onRemove }) => {
6   return (
7     <div className="TodoList">
8       {todos.map(todo => (
9         <TodoListItem todo={todo} key={todo.id} onRemove={onRemove} />
10       ))}
11     </div>
12   );
13 };
14
15 export default TodoList;
16
```

이제 삭제 버튼을 누르면 TodoListItem 에서 onRemove 함수에 현재 자신이 가진 id를 넣어서 삭제 함수를 호출하도록 설정

```
1 import React from 'react';
2 import {
3   MdCheckBoxOutlineBlank,
4   MdCheckBox,
5   MdRemoveCircleOutline,
6 } from 'react-icons/md';
7 import './TodoListItem.scss';
8 import cn from 'classnames';
9
10
11 const TodoListItem = ({ todo, onRemove }) => {
12   const { id, text, checked } = todo;
13   return (
14     <div className="TodoListItem">
15       <div className={cn('checkbox', {checked})}>
16         {checked ? <MdCheckBox /> : <MdCheckBoxOutlineBlank /> }
17         <div className="text">{text}</div>
18       </div>
19       <div className="remove" onClick={() => onRemove(id)}>
20         <MdRemoveCircleOutline />
21       </div>
22     </div>
23   );
24 };
25
26 export default TodoListItem;
```



수정 기능

- 수정 기능도 방금 만든 삭제 기능과 매우 비슷
- onToggle 이라는 함수를 App에 만들고, 해당 함수를 TodoList 컴포넌트에게 props로 넣어준 다음, TodoList를 통해 TodoListItem 까지 전달해 줌

```
47
48 const onToggle = useCallback(
49   id=>{
50     setTodos(
51       todos.map(todo => todo.id === id ? {...todo, checked: !todo.checked} : todo,),
52     );
53   },[todos],
54 )
55
56 return (
57   <TodoTemplate>
58     <TodoInsert onInsert={onInsert} />
59     <TodoList todos={todos} onRemove = {onRemove} onToggle={onToggle} />
60   </TodoTemplate>
61 )
62 };
63
64 export default App;
```

- 위 코드에서는 배열 내장 함수 map을 사용하여 특정 id를 가지고 있는 객체의 checked 값을 반전 시켜 주었음
- 불변성을 유지하면서 특정 배열 원소를 업데이트해야 할 때 이렇게 map을 사용하면 짧은 코드로 쉽게 작성이 가능
- map 함수는 배열을 전체적으로 새로운 형태로 변환하여 새로운 배열을 생성해야 할 때 사용함

```
48 | const onToggle = useCallback(  
49 |   id=>{  
50 |     setTodos(  
51 |       todos.map(todo => todo.id === id ? {...todo, checked: !todo.checked} : todo,)),  
52 |     );  
53 |   },[todos],  
54 | )
```

- `todo.id === id ? ... : ...` 이라는 삼항 연산자는 `todo.id`와 현재 파라미터로 사용된 `id` 값이 같을때 우리가 정해 준 규칙대로 새로운 객체를 생성하지만, `id` 값이 다를 때는 변화를 주지 않고 처음 받아 왔던 상태 그대로 반환 함
- 그렇기 때문에 map 을 사용하여 만든 배열에서 변화가 필요한 원소만 업데이트되고 나머지는 그대로 남아 있게 됨

App 에서 만든 onToggle 함수를 TodoListItem 에서도 호출할 수 있도록 TodoList를 거쳐 TodoListItem에게 전달함

```
1 import React from 'react';
2 import TodoListItem from './TodoListItem';
3 import './TodoList.scss';
4
5 const TodoList = ({ todos, onRemove, onToggle }) => {
6   return (
7     <div className="TodoList">
8       {todos.map(todo => (
9         <TodoListItem todo={todo} key={todo.id} onRemove={onRemove} onToggle={onToggle} />
10       ))}
11     </div>
12   );
13 };
14
15 export default TodoList;
```

TodoListItem.js 수정

```
1 import React from 'react';
2 import {
3   MdCheckBoxOutlineBlank,
4   MdCheckBox,
5   MdRemoveCircleOutline,
6 } from 'react-icons/md';
7 import './TodoListItem.scss';
8 import cn from 'classnames';
9
10
11 const TodoListItem = ({ todo, onRemove, onToggle }) => {
12   const { id, text, checked } = todo;
13   return (
14     <div className="TodoListItem">
15       <div className={cn('checkbox', { checked })} onClick={() => onToggle(id)}>
16         {checked ? <MdCheckBox /> : <MdCheckBoxOutlineBlank />}
17         <div className="text">{text}</div>
18       </div>
19       <div className="remove" onClick={() => onRemove(id)}>
20         <MdRemoveCircleOutline />
21       </div>
22     </div>
23   );
24 };
25
26 export default TodoListItem;
```


일정 관리

할 일을 입력하세요



☒ 리액트 기초 알아보기



☒ 컴포넌트 스타일링 해보기



☐ 일정 관리 앱 만들어 보기

