# 컴포넌트 설계 작성 실습

실습해 볼 TodoList 예시

<http://viewoss.navercorp.com/UIT/markup-guide-react/blob/dev/sample/sample/index.html>

## 저장소 세팅 및 실행

파일 다운로드 이후 압출을 풀어 모듈설치 및 실행을 진행합니다.

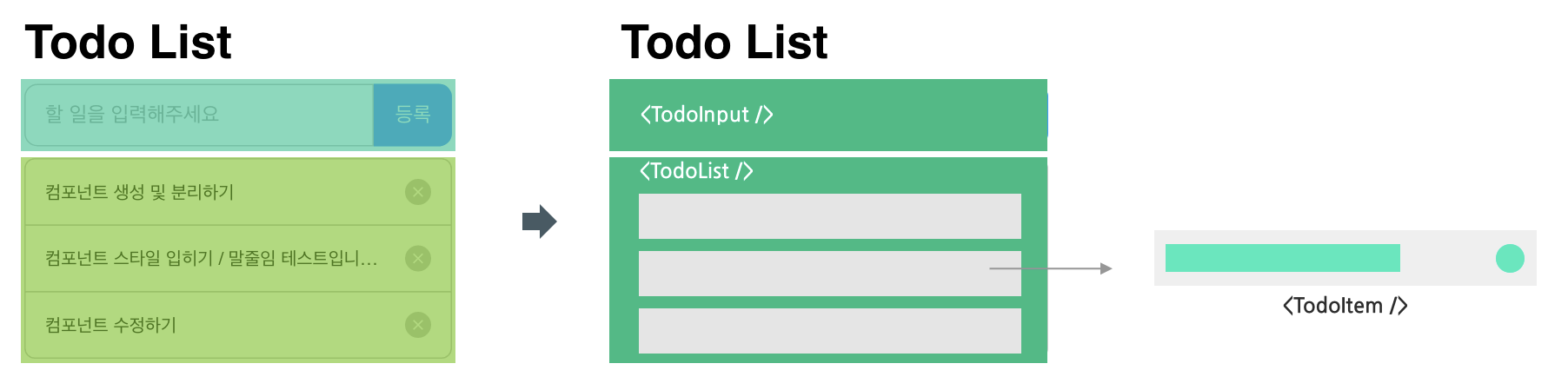
$ npm install  
$ npm run dev

## 컴포넌트 파일 구조

src  
├─── components // 컴포넌트 폴더  
| ├─── TodoContainer.jsx  
| ├─── TodoContainer.scss  
| ├─── ...  
|  
├─── pages // 화면 단위의 React 페이지 컴포넌트 저장 폴더  
└─── scss // 전역으로 사용되는 스타일들(reset, mixin, variables ...)

## 컴포넌트 분리

* 컴포넌트는 기본적으로 UI 사용성을 기준으로 분리합니다. (컴포넌트 설계 및 분리 이론 참고)
* TodoList의 할 일을 입력하는 부분을 <TodoInput/>으로, 할 일 목록을 <TodoList/>으로 분리합니다.



**src/components/TodoInput.jsx 생성**

* <TodoContainer/> 생성한 것과 같이 <TodoInput/> 컴포넌트도 생성합니다.

import React from 'react';  
  
class TodoInput extends React.Component {  
 render() {  
 return (  
 <form>  
 <div className="area\_input">  
 <input type="text" className="input\_todo" placeholder="할 일을 입력해주세요" />  
 <button type="button" className="button\_summit">등록</button>  
 </div>  
 </form>  
 )  
 }  
}  
  
export default TodoInput;

**src/components/TodoList.jsx 생성**

import React from 'react';  
  
class TodoList extends React.Component {  
 render() {  
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 <span className="text">ES6 공부하기</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**src/components/TodoContainer.jsx에서 <TodoList/>, <TodoInput/> import 하기**

* 상단에 TodoInput.jsx와 TodoList.jsx를 import 하고, return문 안에 <TodoInput />, <TodoList />를 불러옵니다.
* 브라우저 화면에 할 일을 입력하는 부분과 할 일 목록을 확인할 수 있습니다.

import React from 'react';  
import TodoInput from './TodoInput'  
import TodoList from './TodoList'  
  
class TodoContainer extends React.Component {  
 render() {  
 return (  
 <div className="container">  
 <h1 className="title">Todo List</h1>  
 <TodoInput />  
 <TodoList />  
 </div>  
 )  
 }  
}  
  
export default TodoContainer;

## 상태값 적용 및 목록의 처리

**step1. 동적으로 바뀌는 값을 변수로 변경**

* 할 일 목록인 <TodoList/>의 span.text 부분은 할 일 텍스트 값이 노출되는 부분입니다. 이 부분은 <TodoInput/>에서 사용자로부터 할 일을 입력받아 동적으로 데이터가 변경될 수 있습니다.
* 이렇게 동적으로 바뀔 수 있는 부분을 변수로 변경해보겠습니다.

AS-IS

import React from 'react';  
  
class TodoList extends React.Component {  
 render() {  
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 <span className="text">ES6 공부하기</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

TO-BE

import React from 'react';  
  
class TodoList extends React.Component {  
 render() {  
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 {/\* 동적으로 바뀔 수 있는 부분을 변수로 변경 \*/}  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step2. props용 변수에 값을 할당**

* step1에서 동적으로 바뀌는 부분을 {todo}로 변경해보았고, 이 todo 변수의 값을 갖고 있을 대상이 필요합니다.
* 이 대상을 DEMO\_PROPS로 설정하고 render 함수 내에 할당해줍니다.

import React from 'react';  
  
class TodoList extends React.Component {  
 render() {  
 // props용 변수에 값을 할당  
 const { todo } = DEMO\_PROPS;  
   
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step3. DEMO\_PROPS에 필요한 값을 정의**

* span.text에 동적으로 변경될 수 있는 할 일 데이터를 DEMO\_PROPS에 임시로 정의해줍니다.
* DEMO\_PROPS는 마크업 개발용 임시 데이터라고 생각하면 됩니다.

import React from 'react';  
  
// 동적 처리에 필요한 데이터  
const DEMO\_PROPS = {  
 todo: 'ES6 공부하기'  
};  
  
class TodoList extends React.Component {  
 render() {  
 const { todo } = DEMO\_PROPS;  
  
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step4. 여러 개의 li 표현**

import React from 'react';  
  
const DEMO\_PROPS = {  
 todo: 'ES6 공부하기'  
};  
  
class TodoList extends React.Component {  
 render() {  
 const { todo } = DEMO\_PROPS;  
  
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step5. 배열형태의 DEMO\_PROPS**

* step4에서는 같은 todo를 노출시켜줬고 이번에는 각각 다른 todo를 적용시켜봅시다.
* 각각 다른 todo를 적용시키기 위해서는 배열 형태의 DEMO\_PROPS를 만들어보겠습니다.

import React from 'react';  
  
// 배열 형태로 DEMO\_PORPS 수정하기  
const DEMO\_PROPS = {  
 itemList: [  
 { todo: 'ES6 공부하기' },  
 { todo: '회의 준비하기' },  
 { todo: '공통 헤더 컴포넌트 생성하기' },  
 { todo: '공통 푸터 컴포넌트 생성하기' },  
 { todo: '홈 마크업 완료하기' }  
 ]  
};  
  
class TodoList extends React.Component {  
 render() {  
 // todo -> itemList로 변경  
 const { itemList } = DEMO\_PROPS;  
  
 return (  
 <ul className="list\_todo">  
 <li className="item\_todo">  
 {/\* 배열값 지정 \*/}  
 <span className="text">{itemList[0].todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{itemList[1].todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{itemList[2].todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{itemList[3].todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 <li className="item\_todo">  
 <span className="text">{itemList[4].todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step6. Array.prototype.map를 이용한 목록 아이템의 반복 처리**

* 목록 아이템을 일일이 쓰는 방법 말고 자바스크립트 내장함수인 Array.prototype.map을 이용하여 목록 아이템을 반복 처리해봅니다.

import React from 'react';  
  
const DEMO\_PROPS = {  
 itemList: [  
 { todo: 'ES6 공부하기' },  
 { todo: '회의 준비하기' },  
 { todo: '공통 헤더 컴포넌트 생성하기' },  
 { todo: '공통 푸터 컴포넌트 생성하기' },  
 { todo: '홈 마크업 완료하기' }  
 ]  
};  
  
class TodoList extends React.Component {  
 render() {  
 const { itemList } = DEMO\_PROPS;  
  
 return (  
 <ul className="list\_todo">  
 {/\* map을 이용하여 필요한 위치에 목록 출력 \*/}  
 {itemList.map((item) => (  
 <li className="item\_todo">  
 <span className="text">{item.todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 ))}  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step7. key값 설정**

* step6에서처럼 목록 아이템을 반복처리를 할 때 key값이 필요합니다.
* key는 글로벌하게 고유할 필요는 없고 형제 사이에서만 고유하면 됩니다.
* 고유한 key 값을 위해 임의로 DEMO\_PROPS에 id 속성을 추가해줍니다.

참고 [key값에 배열의 index를 사용하지 않는 이유](https://reactjs-kr.firebaseapp.com/docs/reconciliation.html#recursing-on-children)

import React from 'react';  
  
const DEMO\_PROPS = {  
 itemList: [  
 {   
 id: 1,  
 todo: 'ES6 공부하기'  
 },  
 {   
 id: 2,  
 todo: '회의 준비하기'  
 },  
 {   
 id: 3,  
 todo: '공통 헤더 컴포넌트 생성하기'  
 },  
 {  
 id: 4,  
 todo: '공통 푸터 컴포넌트 생성하기'  
 },  
 {   
 id: 5,  
 todo: '홈 마크업 완료하기'  
 }  
 ]  
};  
  
class TodoList extends React.Component {  
 render() {  
 const { itemList } = DEMO\_PROPS;  
  
 return (  
 <ul className="list\_todo">  
 {itemList.map((item) => (  
 // key 속성 추가  
 <li className="item\_todo" key={item.id}>  
 <span className="text">{item.todo}</span>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 ))}  
 </ul>  
 )  
 }  
}  
  
export default TodoList;

**step8. 데이터 추가**

* 할 일에 대한 설명 부분을 추가해봅니다.
* DEMO\_PROPS에 description 속성을 추가합니다.
* 기존에는 추가/수정하려면 일일이 요소를 추가해야 했는데 map 매서드를 사용하면 한 번에 수정할 수 있습니다.

import React from 'react';  
  
// description 속성 추가  
const DEMO\_PROPS = {  
 itemList: [  
 {   
 id: 1,  
 todo: 'ES6 공부하기',  
 description: 'var/let 범위에 대한 이해'  
 },  
 {   
 id: 2,  
 todo: '회의 준비하기',  
 description: '설계셔/디자인 확인 필요'  
 },  
 {   
 id: 3,  
 todo: '공통 헤더 컴포넌트 생성하기',  
 description: '공통 범위 확인, 디자인 가이드 확인'  
 },  
 {  
 id: 4,  
 todo: '공통 푸터 컴포넌트 생성하기',  
 description: '공통 범위 확인, 디자인 가이드 확인'  
 },  
 {   
 id: 5,  
 todo: '홈 마크업 완료하기',  
 description: '리스트 작업 완료하기'  
 }  
 ]  
};  
  
class TodoList extends React.Component {  
 render() {  
 const { itemList } = DEMO\_PROPS;  
  
 return (  
 <ul className="list\_todo">  
 {itemList.map((item) => (  
 <li className="item\_todo" key={item.id}>  
 <span className="text">{item.todo}</span>  
 {/\* 할 일에 대한 설명 데이터 반영 \*/}  
 <p>{item.description}</p>  
 <button type="button" className="button\_delete" aria-label="삭제"><i className="icon\_delete" /></button>  
 </li>  
 ))}  
 </ul>  
 )  
 }  
}  
  
export default TodoList;