1. 터미널에서 yarn create react-app 프로젝트명 으로 리액트 프로젝트를 만든다.

2. app.js를 app.jsx로 바꿔준다. -> 컴포넌트가 들어간 파일은 왠만하면 다 jsx로 바꾸기

3. app.js에서 컴포넌트 어떤거 만들지 구상

4. 컴포넌트 파일 만들고, 컴포넌트 생성(InputForm, TodoList), app.js에서 임포트하기

5. 각 컴포넌트에 Function 만들고(return도 넣어주고) export 도 아래 써주기

6. app.jsx에서 todos 만들기 (useState와 uuid 인스톨해주기)

const intialTodoList = [

{

title: "test",

content: "test",

isDone: false,

id: uuidv4(),

},

];

const [todoList, setTodoList] = useState(intialTodoList);

7. 아래 리턴 컴포넌트들에 프롭스 넘겨주기 + 넘겨주면서 TodoList에 isActive 설정해주기

return (

<div>

<InputForm setTodoList={setTodoList}/>

{/\* 하나는 working , Done \*/}

<TodoList isActive={true} todoList={todoList} />

<TodoList isActive={false} todoList={todoList} />

</div>

);

8.( InputForm) 리턴문

return (

<form onSubmit={onSubmitHandler}>

<input type="text" onChange={onChangeTitleHandler} value={title} />

<input type="text" onChange={onChangeContentHandler} value={content} />

<button>추가하기</button>

</form>

);

}

9. ( InputForm) 리턴문

function InputForm({ setTodoList }) {

const [title, setTitle] = useState("");

const [content, setContent] = useState("");

const onChangeTitleHandler = (event) => {

setTitle(event.target.value);

};

const onChangeContentHandler = (event) => {

setContent(event.target.value);

};

const onSubmitHandler = (event) => {

event.preventDefault();

const newTodo = {

title,

content,

isDone: false,

id: uuidv4(),

};

setTodoList((prev) => {

return [...prev, newTodo];

});

// 초기화해주기

setTitle("");

setContent("");

};

10. (TodoList) 안의 내용

function TodoList({ isActive, todoList }) {

return (

<div>

<h4>{isActive === true ? "working" : " Done "}</h4>

{todoList

.filter((item) => item.isDone === !isActive)

.map((item) => {

return (

<div key={item.id} style={{ border: "1px solid black" }}>

제목<div>{item.title}</div>

내용<div>{item.content}</div>

</div>

);

})}

</div>

);

}

export default TodoList;

-------------------------------------------------------------

샘플로 데이터를 하나 넣어보기

12. { title : "test" , contents : "test" , isDone : false, id : ?? }

13. uuid를 사용할건데, 터미널 창에 npm install uuid 검색해서 설치하기

14. import { v4 as uuidv4 } from 'uuid'; 를 임포트해주기

15. { title : "test" , contents : "test" , isDone : false, id : uuidv4() } id 부분에 uuidv4() 적어주기

16. 근데 이 샘플들이 길어지면 가독성이 안즣으니 , 맨 아래애데가 const initialState 로 = [ { } , { } ] 만들어놓고 useStaet(inintialState)를 이렇게 넣어주기

17. TodoList 컴포넌트를 작성할때, 두 개를 불러와야 하는데, 하나는 working일때 하나는 done일때니 , 두 개의 컴포넌트 태그 안에 isActive라고 임의로 만들어놓고, working중일땐 isActive가 true일때로, done일땐 isActive가 false인걸로 넘겨주기

17. app.jsx에서 프롭스로 TodoList를 넘겨주고, TodoList컴포넌트에 와서 return 안에

< > </ > 빈태그 만들어주고,

<h4>{ isActive ===true ? "working" :

"done" }</h4>

{ todoList.filter(item.isDone===!isActive).map((todo)=> { return

<div key = {item.id}>

<h5></h5>

<p></p>

<button>{ isActive === true ? "완료" : "취소"}</button>

<button>삭제</button>

</div>

}}

그 안에서 받아온 todoList를 map으로 돌려주기, 자바스크립트니까 빈태그안에 { } 삽입하고 이 안에서 작업 { todoList.map(()=>{})

18. inputForm 컴포넌트에 가서 작업해주기

return 문 안에

<form onSubmit={submitClick}>

제목 <input type="text">

내용 <input type="text">

<button>추가하기</button>

</form>

onSubmit은 기본적으로 가지고잇는 속성이 새로고침이니

event.preventDefault();를 해주기

18. input태그와 만든 state를 한몸으로 묶어주기 위해 태그에 value={스테이트이름} 을 써준다

input태그에 onChange에 함수를 넣고, onChangeHandler를 넣어주기

onChange이벤트 발생할때

setNewTodo를 업뎃할건데

먼저 newTodo를 하나 만들어주려면

1) const InitialNewTodo 해서 초기값 하나 만들어주고(이거 만들때 uuid필요하니 uuid4() 임포트 해와주기)

2) const [ newTodo, setNewTodo ] = useState(initialNewTodo); 세팅해주기

const ({name, value } = event.target

setNewTodo{...newTodo , [name] : value })

19.

const newTodo = {

title,

contents,

isDone : false,

id : uuid4() \*\*import해주기

}