

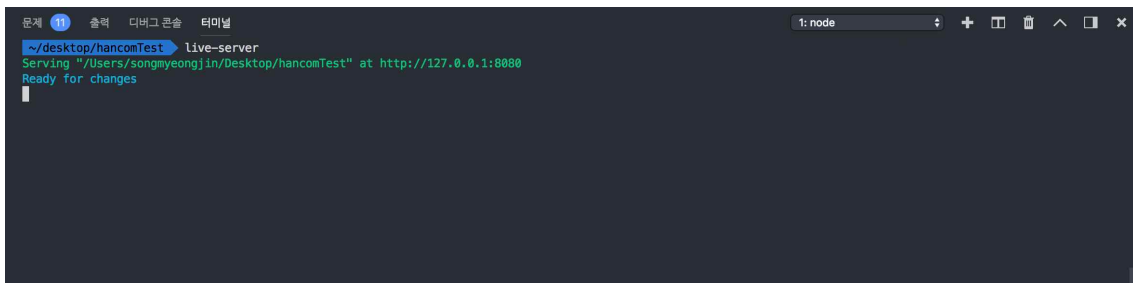
# 미니 ToDo List 구현

클라이언트 직무 송명진

안녕하십니까? 이번에 클라이언트 직무에 지원한 송명진입니다. 제가 개발한 ‘미니 ToDo List 프로젝트’를 정리해서 보내드립니다.

## 1. 준비사항

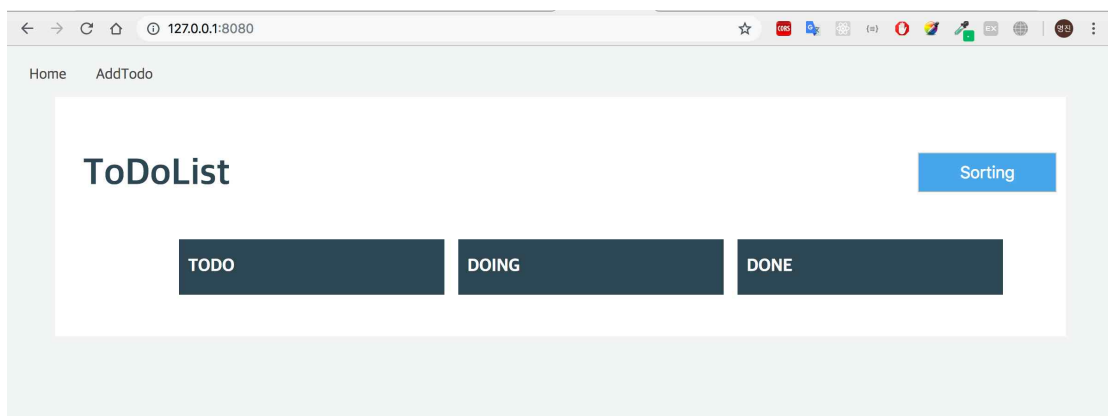
- ✓ 해당 SPA는 크롬 브라우저를 기반으로 작동합니다. 크롬에서 지원하는 LocalDB를 이용하고 있으므로 크롬 브라우저에서 접속해주시길 바랍니다.
- ✓ 해당 기능 중에는 AJAX를 이용하여 데이터를 제공받는 부분이 있습니다. 이를 해결하기 위해서 자바스크립트의 CORS 정책으로 인해 웹 서버를 가동시켜 주셔야 합니다. 저는 npm 라이브러리 중 하나인 live-server라는 명령어를 이용하여 서버를 가동했습니다. <https://www.npmjs.com/package/live-server> 본 링크는 live-server의 설치와 사용 방법이 담겨있는 url입니다. 하단의 방법처럼 수행해주시면 됩니다.



```
~/desktop/hancomTest live-server
Serving "/Users/songmyeongjin/Desktop/hancomTest" at http://127.0.0.1:8080
Ready for changes
```

## 2. 진행과정

1. 해당 todolist의 첫화면입니다. 아무것도 담겨있지 않은 상태입니다. 화면에는 네비게이션 바와 전체 리스트를 보여주는 화면 그리고 id별로 정렬을 해주는 Sorting 버튼으로 구성되어 있습니다.



2. 네비게이션 바의 AddTodo버튼을 누르시면 아래와 같은 화면이 나옵니다. 이전 화면은 home 화면으로 돌아가는 버튼이고 지우기는 폼에 작성된 모든 데이터를 지우는 역할을 합니다. 더불어 제출 버튼은 데이터를 등록하는 버튼입니다. 일단 등록을 해보겠습니다.

127.0.0.1:8080 내용:  
{"id":1,"title":"스터디 하기","category":"Working","name":"송명진","type":"todo"}

확인

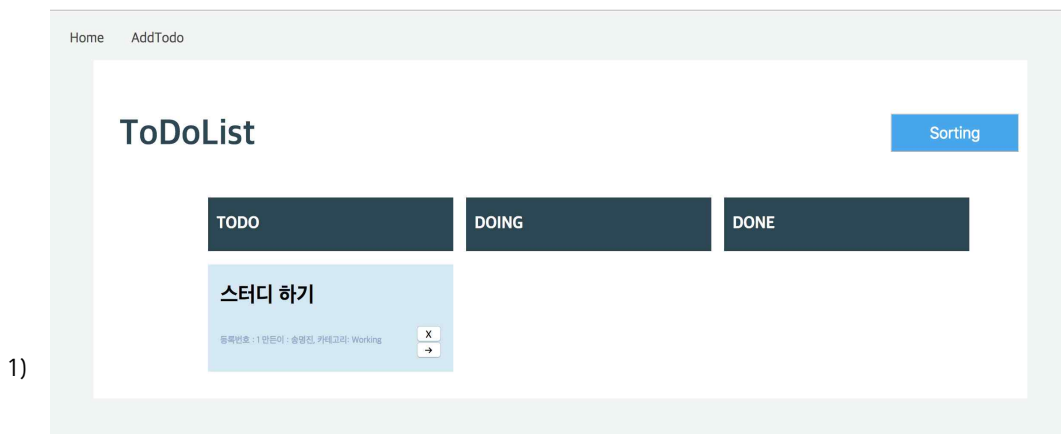
어떤일인가요?  
스터디 하기

누가 할일인가요?  
송명진

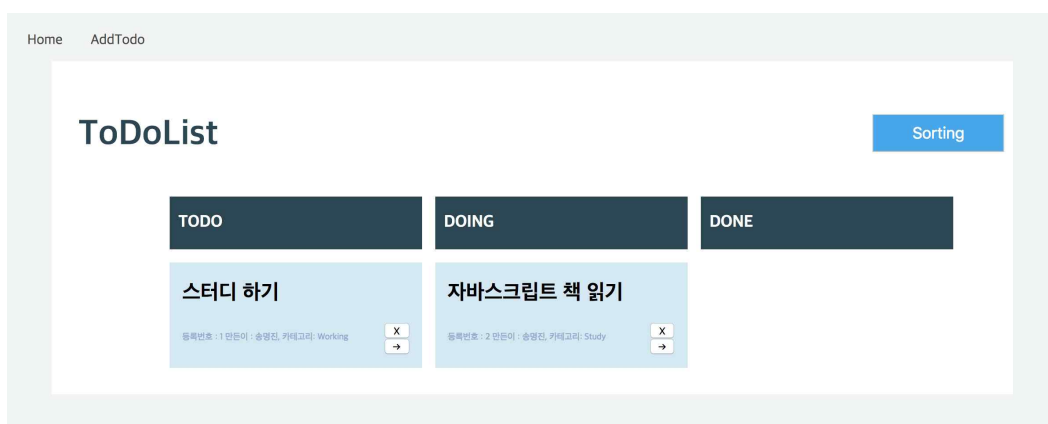
카테고리를 선택하세요  
☐ Study ☐ Exercise ☒ Work

< 이전    내용 지우기    제출

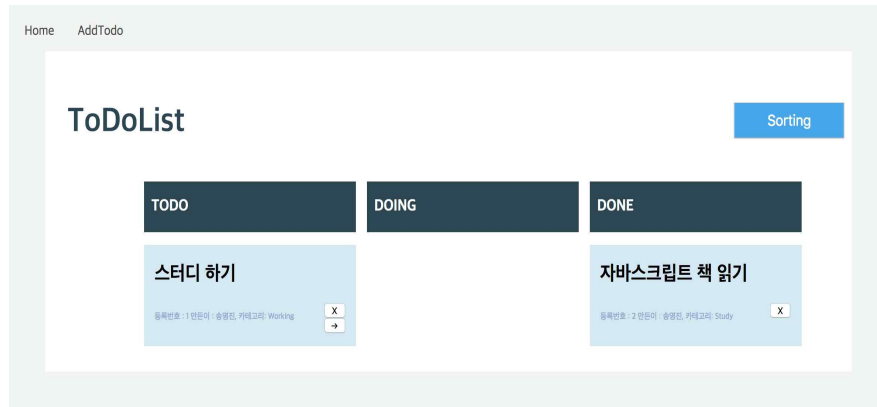
3. 등록 후에는 하단의 그림처럼 카드 형식의 todolist가 저장됩니다. 또한 'X' 버튼을 누르면 삭제되고, '→' 버튼을 누르면 다음 단계로 진행됩니다.



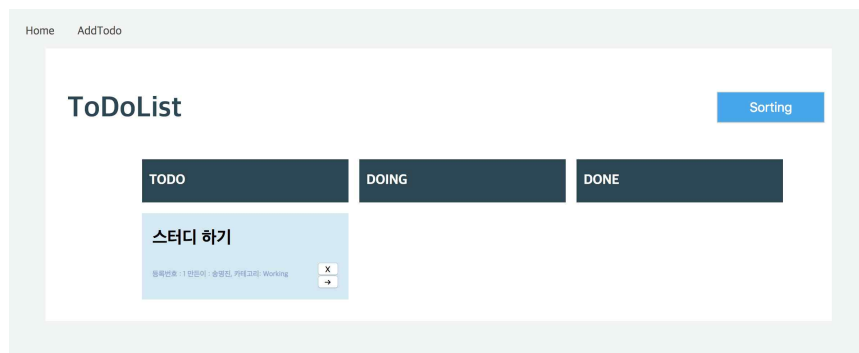
2)



4. Doing 단계까진 '→' 버튼이 사용되지만 Done 상태에는 더 이상 진행할 단계가 없으므로 해당 버튼이 없어집니다.

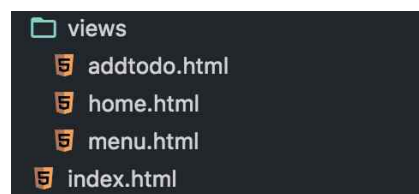
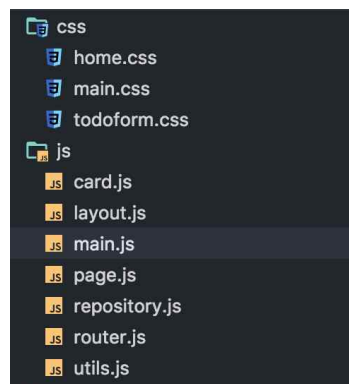


5. 마지막으로 데이터를 삭제해보겠습니다. 'X' 버튼을 누르면 다음과 같이 데이터가 삭제되어서 나옵니다.



### 3. 설계

- ✓ 본 프로젝트는 SPA를 구현하는 프로젝트이므로 데이터의 변동에 따라 렌더링 작업을 하고 라우터를 구현하여 다른 url로 이동할 수 있도록 구현되어 있습니다. 더불어 다른 화면인 라이브러리를 쓰지않고 바닐라 js 형태로 만들어져 있습니다.



- ✓ 파일 구성은 index.html과 css 와js 그리고 template방식으로 화면을 그릴 때 사용할 html을 모은 views 폴더입니다.
- ✓ Card.js 는 todo List를 만드는 함수가 저장되어 있습니다.
- ✓ Layout.js 는 화면 전체 레이아웃을 구성하는 함수로 현재 페이지를 불러오고 (load 함수 이용) 이를 rootElement에 저장하는 역할을 합니다. 이를 통해 화면에 직접 뿌려주는 역할을 합니다.
- ✓ Main.js는 todolist의 데이터 관리를 하는 state와 DB관리를 하는 repository 그리고 라우터 객체를 선언하는 것으로 되어있습니다.
- ✓ Page.js는 화면의 구성을 이루는 객체로 AJAX를 통해 html template을 불러오고 이를 promise를 이용하여 순차적으로 로드하고 버튼 이벤트를 추가하는 역할을 담당합니다.
- ✓ pageName변수를 이용하여 현재 어떤 화면을 렌더링 해야하는지 파악하는 역할을 합니다.
- ✓ Repository.js는 Local DB의 저장과 불러오기 기능을 담당합니다.
- ✓ Router.js는 브라우저의 url의 해시 값을 비교하여 페이지를 구분하고 로드하는 역할을 합니다.
- ✓ Utils.js는 정렬을 담당하는 함수를 담은 객체입니다.

#### 4. 설계 과정

- ✓ Main.js에서 전반적인 구동을 실행합니다.
- ✓ State 객체를 이용하여 React의 State처럼 각 브라우저마다 todoList의 상태를 유지하려고 했지만 페이지가 로드될 때마다 설정된 초기화되는 문제점이 있었습니다.
- ✓ 이를 해결하기 위해 페이지를 새롭게 로드할 때마다 DB의 정보를 로드하는 방식으로 todoList의 상태 관리를 진행했습니다.
- ✓ 라우터 객체를 만들 때 화면의 이름과 Layout에 navbar와 해당 페이지의 데이터를 넣어서 전반적인 렌더링을 라우터에서 관장하게 제작했습니다.
- ✓ 화살표(→) 버튼이나 X버튼을 누르면 state.todoList와 repository의 데이터를 갱신하여 상태 관리를 진행했습니다.