**#React Router**

페이지 처리와 분기

원하는 컴포넌트를 보여주고 원하지 않는 컴포넌트를 보여주지 않는 역할.

리액트 프로젝트 생성 후 리액트 라우터 설치

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

최신버전이 잘 올라간 모습

Src 밑에 router 폴더 생성

라우터에 대한 설정을 모을 폴더임.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

라우터 설정 테스트

멀티미디어 소프트웨어, 소프트웨어, 텍스트, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

설정 후 npm start를 하면 이렇게 나온다.

라우터 설정 시 배열에 아무것도 넣지않아 뜨는 오류.

라우터에 따라 보여주는 컴포넌트를 pages라 하여 만들어보자.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Src 폴더 밑에 MainPage.js를 만들고 단축키 ffc를 통해 아래와 같이 작성해 나간다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

아무것도 없는 빈경로 일 때 보여줄 컴포넌트인 MainPage를 지정

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

컴포넌트가 많아지게 되면 다수의 컴포넌트를 로드한 후에 하나의 컴포넌트를 로드하기 때문에 느려진다.

이 때문에 코드 스플릿팅을 이용한다.

#라우터 코드 변경

기존

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

변경

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

컴포넌트 로딩이 되기 전에만 잠깐 Loading이라는 컴포넌트를 잠시 보여줌. 이것을 코드 스플리팅이라고함.

텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

AboutPage를 추가해보자

텍스트, 스크린샷, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이렇게 하면 문제가 있다.

페이지 이동을 통한 처리가 되긴하지만 이동 시 매번 리액트가 로드가 된다는 문제가 있다.

같은 이유로 리액트에서는 a태그가 아닌 링크를 사용한다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

마찬가지로 about 페이지에서도 링크를 하나 추가해보자

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

이후 메인 페이지에서 About을 클릭하기 전후로 네트워크 탭의 변화를 관찰하면

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명전 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명후

링크 사용을 통해서 한 어플리케이션에서 화면이 전환이 되면서 그때그때 필요한 것들을 가져온다.

이것을 리액트 코드 스플릿팅이라고 한다.

**#레이아웃 만들기**

레이아웃 별로 화면을 구성하기 위해 새로운 layouts 디렉토리를 추가한다.

BasicLayout을 작성한 후 이것을 main페이지에 적용시켜보자.

Props 속성 중 children을 활용해서 해당 레이아웃이 적용된 페이지에 맞게 동적으로 작성할 수 있다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

div태그 부분이 작성한 레이아웃에서 children자리에 들어가게 되는 것이다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 about 페이지도 basiclayout을 적용시켜보자.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

헤더와 사이드바는 그대로인데 children 속성을 활용해 작성한 개별 컴포넌트만 변경되어 나온다.

이런식으로 공통 레이아웃이 적용된 상태에서 내용물만 채워주는 구조로 할 수 있다.

**#상단 메뉴 컴포넌트 만들기**

**컴포넌트 폴더를 만들고 메뉴 폴더를 만들고 BasicMenu.js 를 작성해준다.**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**만든 메뉴 컴포넌트를 BasicLayout에가서 import를 한 후 적용 시켜보자.**

**기존 header 부분을 삭제하고 베이직메뉴로 갖다 쓰기.**

**스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명**

**#앱 개발 TodoList 구성하기**

**텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명**

**라우터 설정부터 추가해야한다.**

우선 메뉴 컴포넌트에 Todo 항목을 추가한다.

추가는 됐어도 현재 경로에 대한 컴포넌트는 없다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

지금부터 Todo내에서만 사용하는 레이아웃을 만들어보자. (outlet기술을 활용해서)

즉 outlet기술을 활용해서 서브 메뉴를 잡아보자.

일단 책에 코드를 따라 BasicLayout.js 를 수정한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Todo에 관한 목록과 페이지를 구성을 모아둘 폴더 todo를 생성

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이제 서브 메뉴로 사용할 ListPage를 작성해보자

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

작성한 리스트 페이지에 대해서 라우터 설정을 해주는데

Todo의 IndexPage에서 사용했던 Outet에 리스트 컴포넌트를 삽입하기 위해 children이라는 개념을 사용해서 작성해준다.

이렇게 하면 todo 항목 중 리스트를 타고 들어갔을 때 즉

<http://localhost:3000/todo/list>에 들어가게 되면 Todo의 IndexPage작성의 Outlet 자리에 list컴포넌트가 자리하게 된다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명