



'React Router Dom'

항해99 14기 이준영

1. **Routing**이란 무엇인가?

■ Routing이란 무엇인가?

Rounting이란?

: Rounting : 경로 선택, 경로 결정

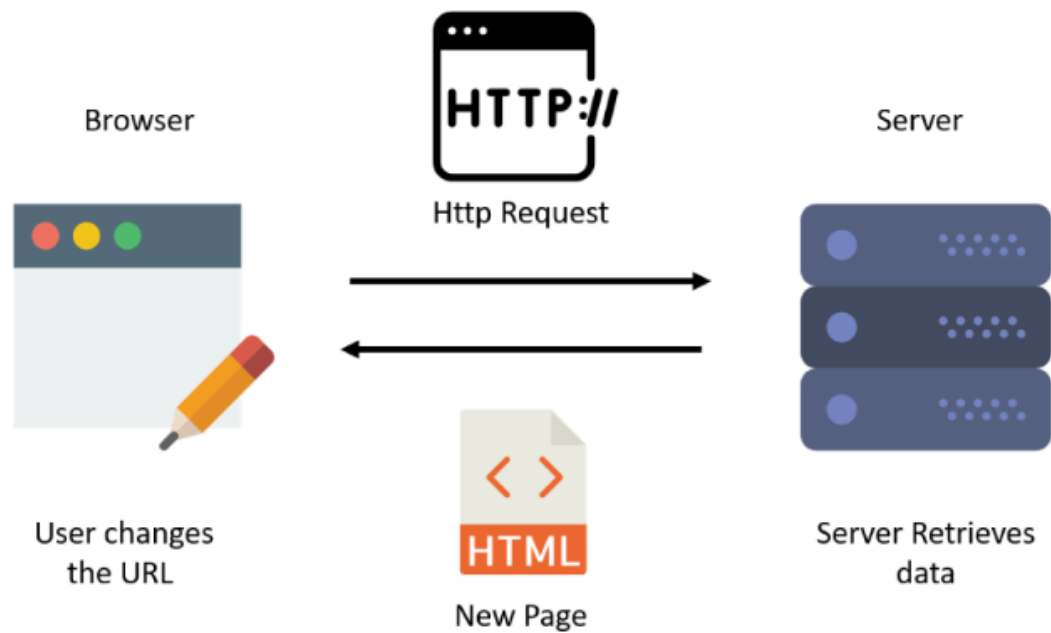
: 해당하는 페이지로 이동하기 위해 새로운 URL로 서버에 요청을 하면 그 응답으로 data를 받아 새로운 페이지를 보여주는 것,
즉 해당 경로의 페이지로 이동하는 것을 의미한다.



Routing

일반적인 Routing 과정

서버에 URL로 새로운 페이지 전체(html, css, js, img 등)를 요청해서 응답 받은 새 페이지를 업데이트 한다.



■ CSR(Client Side Routing) - React Router

서버에 URL로 새로운 페이지를 요청하는 것이 아니라 페이지는 유지하되, Application에서 업데이트가 필요한 부분만, 경로에 해당하는 컴포넌트만 부분적으로 업데이트하고, 부분적으로 네트워크 통신(fetch 등)하여 JSON 형태 data를 받아온다.



CSR(Client Side Routing) - React Router

React Router는 "클라이언트 측 라우팅"을 가능하게 합니다.

기존 웹 사이트에서 브라우저는 웹 서버에서 문서를 요청하고 CSS 및 JavaScript를 응답받아 서버에서 보낸 HTML을 렌더링합니다. 사용자가 링크를 클릭하면 새 페이지에 대한 프로세스가 다시 시작됩니다.

클라이언트 측 라우팅을 통해 앱은 서버에서 다른 문서를 다시 요청하지 않고 링크 클릭에서 URL을 업데이트할 수 있습니다. 대신, 앱은 일부 새 UI를 즉시 렌더링하고 네트워크 통신하여(fetch) 새 정보로 페이지를 업데이트할 수 있습니다.

이는 브라우저가 전체 새로운 문서를 요청하거나 다음 페이지를 위해 CSS 및 JavaScript를 가져올 필요가 없기 때문에 더 빠른 사용자 경험을 가능하게 합니다. 또한 애니메이션과 같은 것으로 보다 동적인 사용자 경험을 가능하게 합니다. - <React Router 공식문서>

=> **SPA(Single Page Application)의 장점 = '부드러운 사용자 경험성'을 유지하면서,**

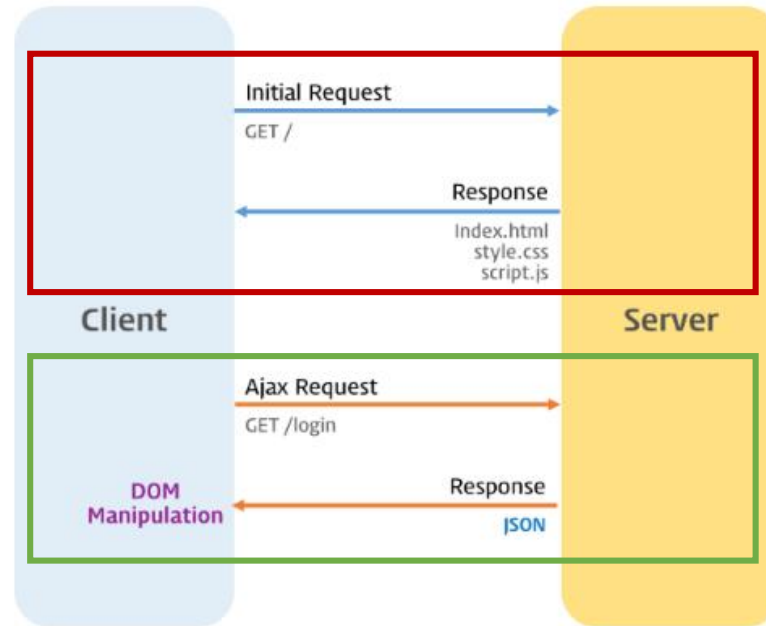
MPA의 장점 = '멀티 페이지'도 그대로 사용할 수 있다.

2. 일반적인 Routing과

Client Side Routing의 차이점

MPA vs SPA(React Router)

MPA
= 일반적인 Routing



SPA
= CSR(React Router)

* Request/Response는 파일 단위로 실시된다. 즉 html, css, js 파일은 한번에 Request/Response되는 것이 아니라 각각의 파일 단위로 Request/Response된다.

Ajax Lifecycle

■ React router의 장점

Client가 모든 페이지를 가지고 있기 때문에 서버에 페이지 요청은 처음 1회만 발생한다.

= 이후 페이지 이동에서 전체 페이지를 업데이트 하거나 하지 않아 새로 고침에 의한 깜빡임이 발생하지 않고,
필요한 부분만(컴포넌트 등) 부분적으로 업데이트하기 때문에 사용자 경험성이 앱과 같이 빠르고 부드럽다.

3. 간단한 **React Router** 사용 방법

React router 사용 방법

1) 설치

```
yarn add react-router-dom
```

2) Router.js 파일에서 모듈화

```
const Router = () => {  
  return (  
    <BrowserRouter>  
      <Routes>  
        <Route path="/" element={<Home />}></Route>  
        <Route path="about" element={<About />}></Route>  
      </Routes>  
    </BrowserRouter>  
  );  
};  
  
export default Router;
```

<Route>에서 path="/"에 경로 설정, 해당 경로에서 보여줄 컴포넌트 전달

React router 사용 방법

3) App.js에서 import하여 사용

```
import Router from "shared/Router";

function App() {
  return (
    <Router>
      <div>APP</div>
    </Router>
  );
}
```

4. React Router Hooks

React router Hooks - useNavigator

1) useNavigator : 조건이나 연산이 필요한 곳에서 경로를 이동할 수 있다.

navigator() 함수 호출 하여 ("/path")안에 경로를 설정한다.

ex) 로그인 페이지에서 아이디와 비밀번호를 입력하고 로그인 버튼을 클릭했을 때 다른 페이지로 이동할 때

```
const navigator = useNavigate();  
  
<button onClick={() => navigator("/main")}> 로그인 </button>
```

💡 useNavigate를 직접 호출하지 않고 변수로 선언해서 반환된 함수를 호출하는 이유

- hook의 규칙

컴포넌트 최상단에서만 호출해야한다.

다른 조건문, 함수선언문, 반복문 등 중첩된 함수 안에서는 호출 할 수 없다.

컴포넌트 내부 함수에서 callback으로도 호출 할 수 없다.

함수 컴포넌트에서 호출할 수 있다. (Custom hook에서 예외처리 가능)

React router Hooks - Link

2) Link: click시 이동하는 URL을 지정하면, 해당 경로로 이동

하이퍼링크 형식, 클릭만 하면 이동하기 때문에 다른 연산과정 없이 페이지를 이동할 때 쓰인다.

```
<Link to="/">Home</Link>
```

a태그를 대체하여 사용 => 새로 고침, 모든 컴포넌트를 다시 렌더링, state 초기화 => 불필요한 렌더링으로 성능에 악영향

💡 Link 컴포넌트와 <a>의 차이점

는 전체 페이지를 재렌더링 시킨다.
페이지 새로 고침, 모든 컴포넌트를 다시 렌더링, state 초기화
=> 불필요한 렌더링으로 성능에 악영향을 끼친다.

<Link />는 SPA의 특징에 맞게 필요한 부분만 재렌더링하고 나머지 부분은 그대로 유지된다.
데이터를 필요한 부분만 불러올 수 있기 때문에 속도향상에 도움이 된다.

- 쓰임의 차이

 는 외부 프로젝트와 연결 할 때 주로 사용한다.
<Link />는 프로젝트 내에서 페이지 전환하는 경우 사용한다.

React router Hooks - useLocation

3) useLocation : 현재 위치(경로)에 대한 정보를 알려준다.

localhost:3001/about

```
const location = useLocation();  
console.log(location);
```

```
About.jsx:7  
▶ {pathname: '/about', search: '', hash: '', state: null, key: 'default'}
```

- pathname : /about 이 출력

- search : ?keyword=react => 쿼리 스트링이 출력된다.

React router Hooks - useParams

localhost:3000/works/1

4) useParams : 현재 routing 페이지로 넘어온 params들의 정보를 얻을 수 있다.

```
▶ {id: '2'} work.jsx:16
▶ {id: '3'} work.jsx:16
▶ {id: '1'} work.jsx:16
▶ {id: '4'} work.jsx:16
```

- path의 parameter의 정보를 얻을 수 있다. => 동적 라우팅에서 사용

5. 동적 Routing

동적 Routing

동적 라우팅 : path를 직접 입력하지 않아도, 유동적인 값(parameter)을 넣어서 특정 페이지로 이동하게끔 구현하는 방법을 말한다.

1) 만약 /works라는 경로의 하위 경로로 1~n 까지를 만들어 정보를 각각 다르게 주고싶을 경우,

```
<Route path="works/1" element={<Work />} />
<Route path="works/2" element={<Work />} />
<Route path="works/3" element={<Work />} />
...
```

이렇게 직접 일일이 <Route>를 전부 설정할 수 없기 때문에 다음과 같이 동적 라우팅을 설정한다.

```
<Route path="works" element={<Works />} />
  {/* 아래 코드를 추가해주세요. 📌 */}
<Route path="works/:id" element={<Work />} />
```

<Rooms> 컴포넌트에서 상세 페이지를 동적 라우팅

Click!

어디든지 | 언제든 일주일 | 게스트 추가

당신의 공간을 에어비앤비하세요

최고의 전망
 한옥
 기상천외한 숙소
 캐슬
 동굴
 개인실
 해변 바로 앞
 한적한 시골
 인기 급상승
 국립공원
 멋진 수영장
 료칸
 저택
 캠핑장
 초소형 주택
 필터

파사이(Pasay), 필리핀

2,628km 거리

5월 31일 ~ 6월 5일

₩65,240 /박

Lubao, 필리핀

2,592km 거리

5월 4일~10일

₩404,190 /박

Son Trà, 베트남

3,016km 거리

7월 7일~13일

₩147,976 /박

Lipa, 필리핀

2,690km 거리

5월 1일~6일

₩371,880 /박

파사이(Pasay), 필리핀

2,629km 거리

5월 1일~8일

₩70,591 /박

Thành phố Hội An, 베트남

3,028km 거리

5월 5일~10일

₩485

Thành phố Hội An, 베트남

3,028km 거리

5월 2일~7일

₩483

Dolsan-e

326km 거리

5월 22일~28일

지도 표시하기

Bayan ng Silang, 필리핀

2,668km 거리

5월 7일~13일

₩493





Son Trà, 베트남


3,016km 거리

5월 1일~6일



<Room> 컴포넌트 상세페이지 => 동적 라우팅으로 같은 페이지에서 다른 정보들 보여주기






airbnb.co.kr/rooms/51285965?adults=1&category_tag=Tag%3A677&children=0&enable_m3_private_room=false&infants=0&pets=0&...

 검색 시작하기  당신의 공간을 에어비엔비하세요  

 펜트하우스가 내려다보이는 마닐라 베이 스위트룸 + 수영장 1BR


★ 4.67 · 후기 136개 · 파사이(Pasay) · 메트로 마닐라 · 필리핀


 공유하기  저장




Rence 님이 호스팅하는 레지던스 전체

최대 인원 4명 · 침실 1개 · 침대 1개 · 욕실 1.5개



 셀프 체크인

열쇠 보관함을 이용해 체크인하세요.

 마음껏 물놀이를 즐기세요

해당 지역에서 수영장을 갖춘 몇 안 되는 숙소 중 하나입니다.

₩62,405 ₩55,280 /박

★ 4.67 · 후기 136개

체크인 2023. 5. 31.	체크아웃 2023. 6. 5.
인원 게스트 1명	▼

4-3) 다음과 같이 동적 라우팅 활용해 보기

```
//App.js
<BrowserRouter>
  <Routes>
    <Route path="rooms/:roomId" element={<Room />} />
    ...
  </Routes>
</BrowserRouter>

// Rooms.js
import { Link } from "react-router-dom";
function Rooms() {
  return(
    <Link to={` /rooms/${data.roomId}`}>
      id번째 숙소로 이동!!
    </Link>
  )
}
export default Rooms;

// Room.js
import React from 'react';
import { useParams } from 'react-router-dom';

function Room() {
  const { roomId } = useParams();
  return (
    <div>
      {roomId}번째 숙소 상세페이지입니다.
    </div>;
  )
}

export default Room;
```

이와 같이 동적 라우팅을 이용하면 id값에 따라 상세페이지가 변경되게 만들고, 해당 하는 상세페이지의 useParams를 이용해 id값에 해당하는 data를 보여주도록 하면 모든 상세 페이지들을 일일이 만들지 않아도 쉽게 구현 할 수 있다.

6. 중첩 Routing

중첩 Routing : 비슷한 디자인의 페이지에서 내용만 변경하고 싶을 경우

<AccountPage>

로그인 | 회원가입

SAMSUNG Account

<Login>

삼성계정으로 로그인

이메일 주소 또는 전화번호

☐ ID 기억하기

다음

ID 찾기

계정 생성



Google 계정으로 로그인



QR 코드로 로그인

SAMSUNG Account

<Signup>

삼성계정 하나면 충분합니다.

한번의 가입으로 언제 어디서나 삼성 서비스를 즐기실 수 있습니다.

만 14세 이상 회원 가입

만 14세 미만이라면?

만 14세 미만은 보호자(법정 대리인)와 함께 가입해 주십시오.

만 14세 미만 회원 가입

중첩 Routing

1. 비슷한 디자인의 content-box를 재사용하기 위하여 AccountPage 컴포넌트를 생성하고, 하위 경로로 /account/login, /account/account로 설정

AccountPage.jsx

```
export const AccountPage = () => {  
  return (  
    <>  
      <div className={styles.accountBox}>  
        <Link to='/'>  
          <img className={styles.logo} src={HomeLogo} />  
        </Link>  
      </div>  
    </>  
  );  
};
```

컴포넌트 골격과 내부 logo 부분까지의 content-box 디자인은 Login과 Signup 페이지에서 모두 동일하여 AccountPage를 생성하여 재사용 가능하도록 하였고, 이제 페이지 내부에서 하위 페이지를 연결하는 작업이 필요

중첩 Routing

2. App.jsx 중첩 라우팅

AccountPage 내부에 중첩 라우팅(nested routing)하여 각각 Route path를 '/account/LoginPage', '/account/signupPage' 로 지정하면 간단하게 중첩라우팅을 사용할 수 있다.

AccountPage.jsx

```
<RecoilRoot>
  <BrowserRouter>
    <Routes>
      <Route path='/account' element={<AccountPage />}>
        <Route path='/account/login' element={<LoginPage />} />
        <Route path='/account/sign-up' element={<SignupPage />} />
      </Route>
    </Routes>
  </BrowserRouter>
</RecoilRoot>
```

중첩 Routing

3. Outlet 설정

중첩라우팅을 사용할 부모 컴포넌트(AccountPage)에서 자식 라우트 컴포넌트의 위치를 지정해주어야 URL이 정상적으로 작동한다.

<Outlet/>을 import 하여 부모컴포넌트 내에서 자식 컴포넌트의 라우트할 위치를 지정해 준다.

이 자리에 바로 자식컴포넌트가 위치 할 것이다.

```
export const AccountPage = () => {
  return (
    <>
      <div className={styles.accountBox}>
        <Link to='/'>
          <img className={styles.logo} src={HomeLogo} />
        </Link>
        //자식 컴포넌트의 위치 지정 Outlet 사용
        <Outlet />
      </div>
    </>
  );
};
```

이렇게 자식 컴포넌트의 위치까지 설정해 주면, URL이 정상적으로 이동하는 것을 확인할 수 있다.

감사합니다.