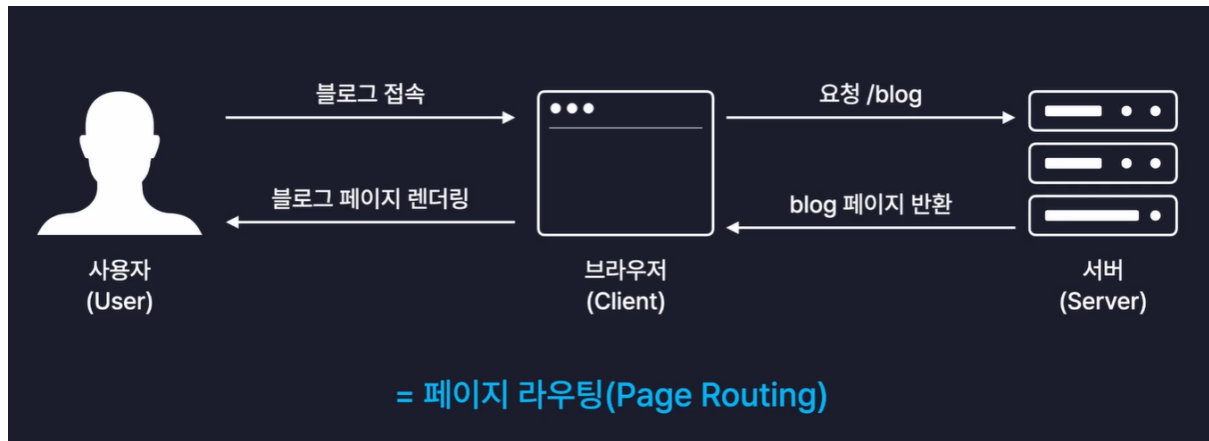


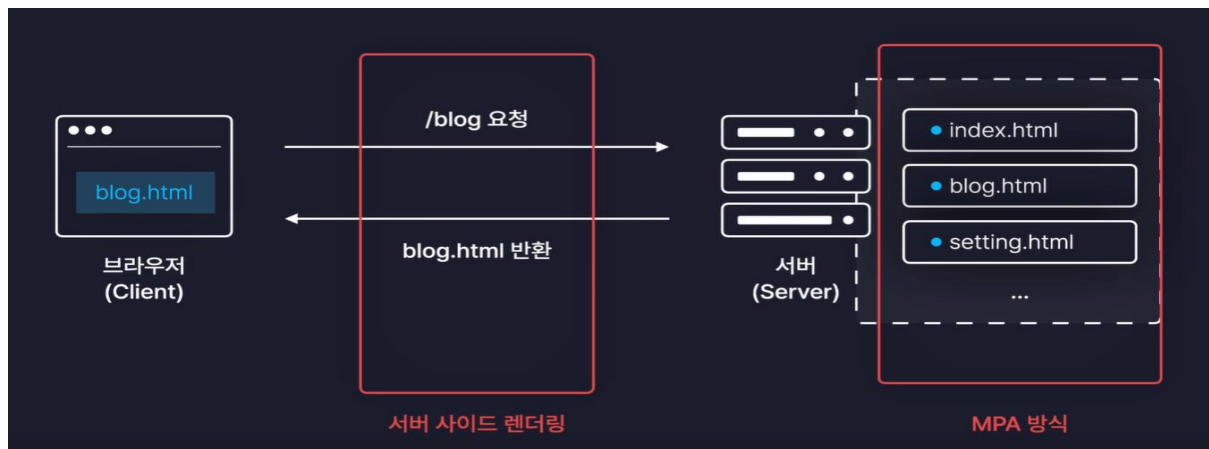
## 페이지 라우팅이란?

경로에 따라 알맞은 페이지로 렌더링하는 과정



### 👉 페이지 렌더링 방식

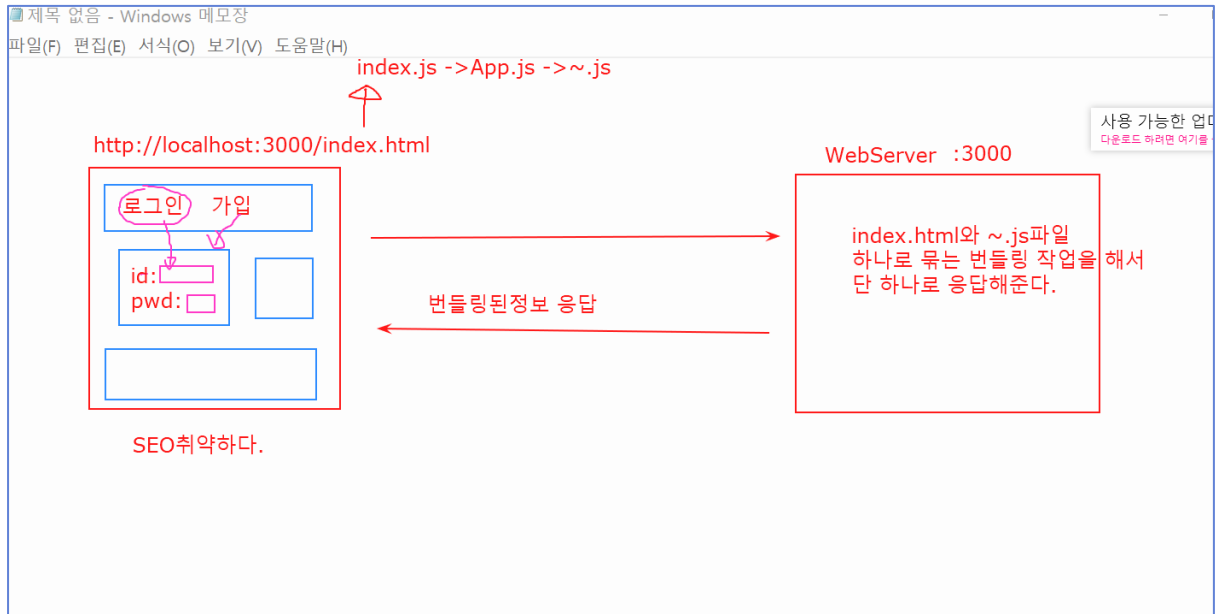
#### 1) MPA(Multi Page Application)



#### 2) SPA(Single Page Application)



React는 SPA로 동작한다.



Routing관련 기술이 많이 있지만 그 중에서도 가장 많이 사용하는 기술이 React Router다.

<https://www.npmjs.com/> 에 등록되어 있는 라이브러리이다.

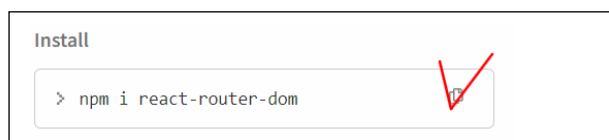
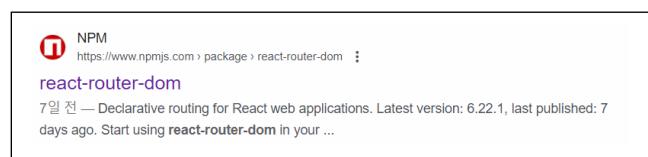
원래 Anchor 태그를 사용해서 다른 페이지로 이동할 수 도 있지만 **Anchor 태그는 화면 전체가 새로고침 되면서 새 페이지로 이동하는 반면에 React Router는 화면 전체가 갱신되는 일 없이 기존페이지 안에서 데이터만 가져오는 방법으로 화면을 갱신해 주기 때문에 불필요한 렌더링이 일어나지 않는다.**

React Router는 버전 5, 버전 6은 사용법이나 코드상 많은 변화가 있다. 현재 최신 버전 6버전을 사용한다. 5버전에 비해서 사용법은 훨씬 간단 해졌다.

## 1. 설치

구글 < react router dom

< 설치 명령어 복사,설치



```

PS C:\Edu\React\ReactWork\step16-router> npm i react-router-dom

added 3 packages, and audited 252 packages in 2s

102 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Edu\React\ReactWork\step16-router> 

```

설치가 제대로 되었는지는 package.json 파일에서 확인할 수 있다.

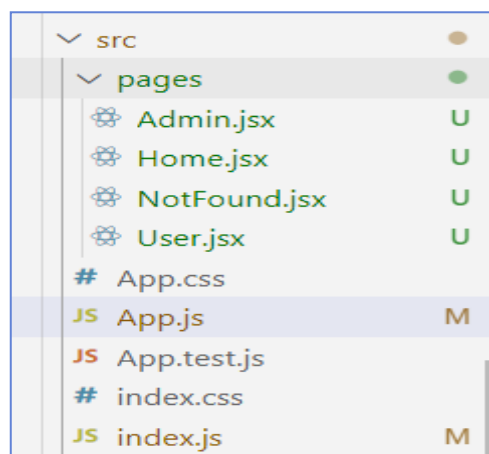
```

step16-router > {} package.json > ...
  9      "lint": "eslint .",
 10      "preview": "vite preview"
 11    },
 12    "dependencies": {
 13      "react": "^18.3.1",
 14      "react-dom": "^18.3.1",
 15      "react-router-dom": "^6.28.0"
 16    },
 17    "devDependencies": {
 18      "@eslint/js": "^8.12.0"

```

## 실습

1. "/" : Home 페이지
2. "/user" : User페이지
3. "/admin" : Admin페이지
4. "/없는주소" : Notfound페이지



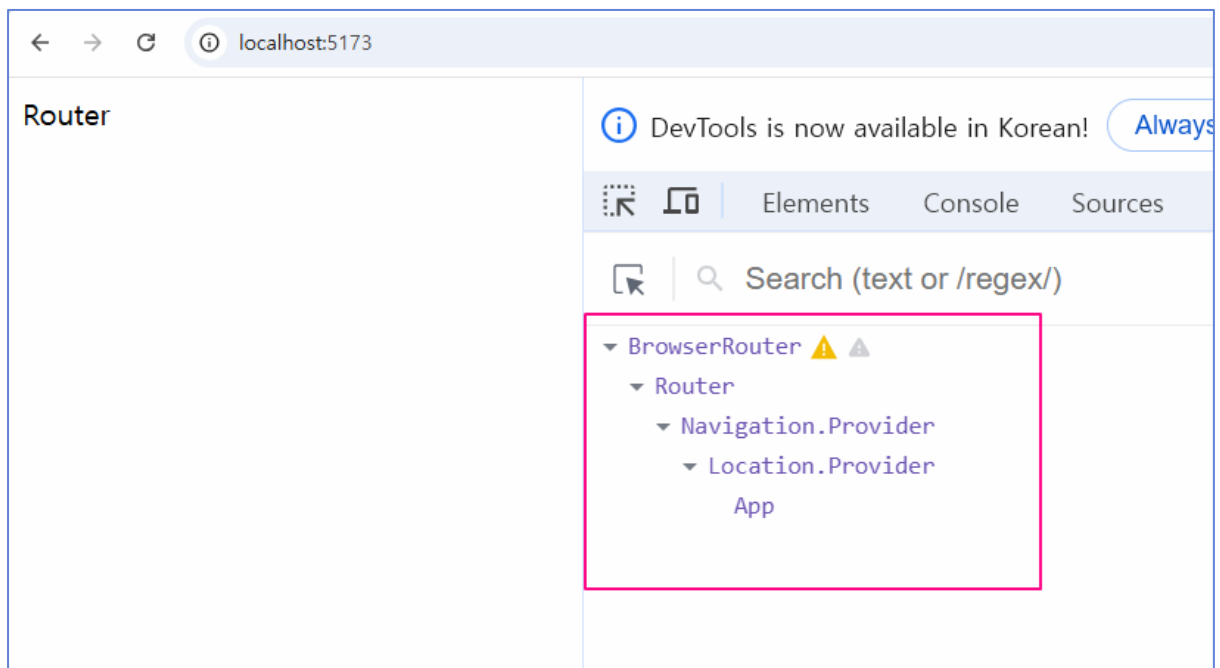
**main.jsx**

step16-router &gt; src &gt; main.jsx

```
1
2 import { createRoot } from 'react-dom/client'
3 import './index.css'
4 import App from './App.jsx'
5 import { BrowserRouter } from 'react-router-dom';
6
7 createRoot(document.getElementById('root')).render(
8   <BrowserRouter>
9     <App />
10  </BrowserRouter>
11
12 )
13
```

→ App을 BrowserRouter로 감싸게 되면  
모든 페이지들이 네비게이터를 공급받게 된다.

실행화면에서 F12 > Components tab 클릭



```

step04-router > src > JS App.js > App
1
2 import './App.css';
3 import {Routes, Route} from "react-router-dom";
4 import Home from './pages/Home';
5 import User from './pages/User';
6 import Admin from './pages/Admin';
7 import NotFound from './pages/NotFound';
8 function App() {
9   return (
10     <Routes>
11       <Route path="/" element={<Home/>}/>
12       <Route path="/user" element={<User/>}/>
13       <Route path="/admin" element={<Admin/>}/>
14       <Route path="/*" element={<NotFound/>}/>
15     </Routes>
16   );
17 }
18
19 export default App;
20

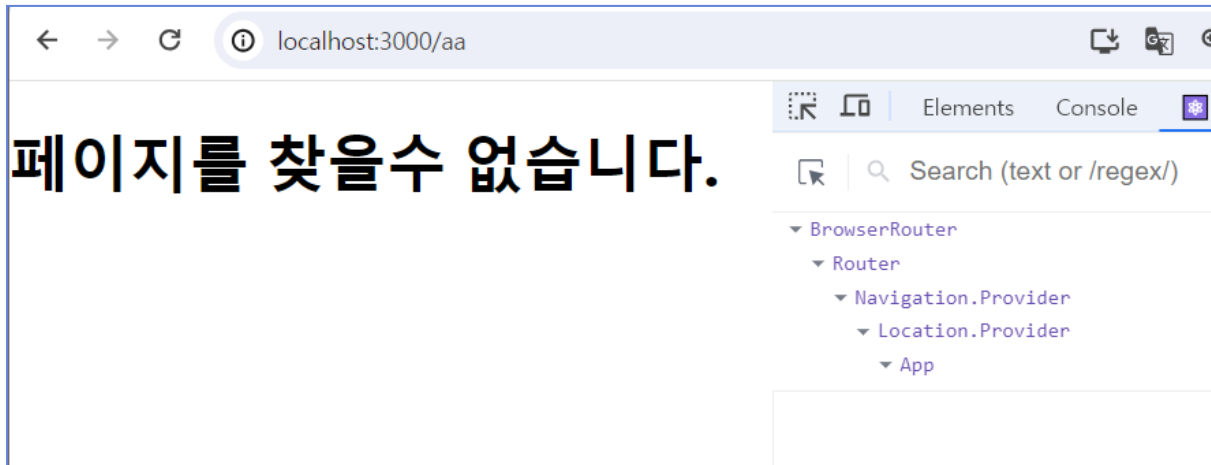
```

→ path속성에 해당하는 주소가 요청이되면 element속성의 컴포넌트를 찾아서 렌더링한다.

BrowserRouter < Routes < Route 순으로 계층구조가 만들어진다.

Route를 위에서 순서대로 찾는다. 맨 마지막에 /\*를 선언 함으로써 이외의 모든 요청은 NotFound 컴포넌트 페이지가 열리도록 한다.

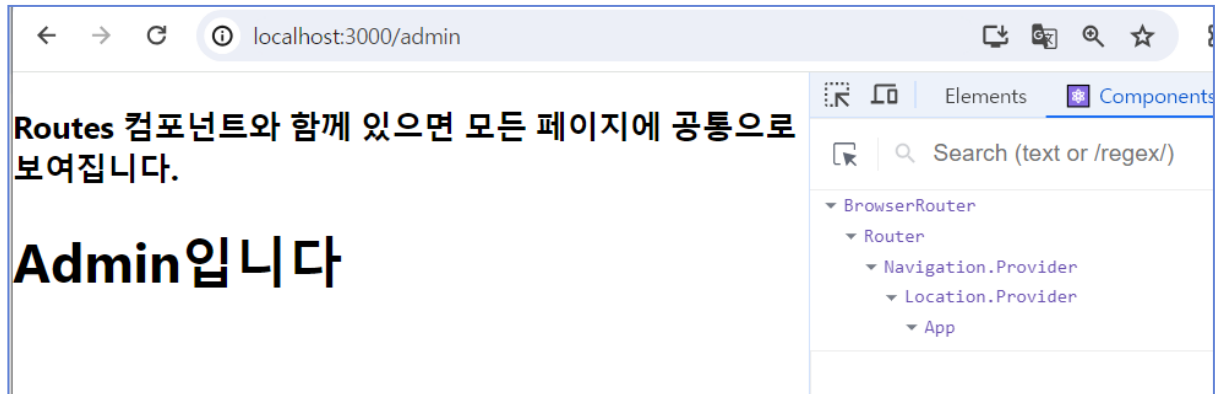




## 👉 알아두기

- 1.<Routes> 컴포넌트 안에는 반드시 <Route> 컴포넌트만 선언 가능 한다.
- 2.<Routes>가 있는 파일에 다른 element가 있으면 항상 함께 출력된다.  
주로 모든 페이지에 공통으로 사용하는 UI를 포함한다.

```
1  import './App.css';
2  import {Routes, Route} from "react-router-dom";
3  import Home from './pages/Home';
4  import User from './pages/User';
5  import Admin from './pages/Admin';
6  import NotFound from './pages/NotFound';
7  function App() {
8    return (
9      <div>
10       <h3>Routes 컴포넌트와 함께 있으면 모든 페이지에 공통으로 보여집니다.</h3>
11       <Routes>
12         <Route path="/" element={<Home/>}/>
13         <Route path="/user" element={<User/>}/>
14         <Route path="/admin" element={<Admin/>}/>
15         <Route path="/*" element={<NotFound/>}/>
16       </Routes>
17     </div>
18   );
19 }
20
```



## 👉 컴포넌트에서 버튼이나 링크를 통해서 페이지 이동하기

<Link>태크는 html의 a 태그를 대체하는 기술.

Link와 A 태그의 차이점?

보여지는 화면은 동일하지만 페이지가 새로고침 되느냐의 여부이다.





```

▼ <div> == $0
  <a href="/">Home</a>
  <a href="/user">User</a>
  <a href="/admin">Admin</a>
  <hr>
  <a href="/">Home</a>
  <a href="/user">User</a>
  <a href="admin">Admin</a>
</div>

```

## 👁 버튼을 클릭했을때나 이벤트가 발생했을 때 특정 컴포넌트 렌더링

: react의 useNavigate라는 Hook을 이용한다.

```

import {Routes, Route, Link, useNavigate} from "react-router-dom";
function App() {
  const nav = useNavigate();
  const btnClick = ()=>{
    nav("/user");
  }
  const btnClick2 = ()=>{
    nav("/admin");
  }
  return (
    <div>
      <div>
        <button onClick={btnClick}>User이동</button>
        <button onClick={btnClick2}>Admin이동</button>

```

## 👁 동적 경로로 페이지 라우팅하기

동적경로란?

: 동적인 데이터를 포함하고 있는 경로를 말한다.



## URL Parameter

/ 뒤에 아이템의 id를 명시

~/product/1

~/product/2

~/product/3

아이템의 id 등의 변경되지 않는 값을  
주소로 명시하기 위해 사용 됨

## Query String

? 뒤에 변수명과 값 명시

~/search?q=검색어

검색어 등의 자주 변경되는 값을  
주소로 명시하기 위해 사용된다

실습

예) /user/1 | /user/2 | /user/3 ....

```
<Route path='/user/:id' element={<User/>}/>
```

→ 경로에 url-parameter를 /:이름 표시  
해줘야한다.

User.jsx에서 전달된 parameter를 어떻게 받아올까?

: react의 useParams hook을 이용한다.

The screenshot shows a web application running on localhost:3000/user/2. The browser address bar is highlighted with a pink box. The page displays "User : 2회원정보입니다." (User : 2 member information). The React DevTools console shows the props passed to the User component: {id: '2'}. The code in the editor shows the useParams hook being used to retrieve the id from the URL.

```
1 import React from 'react';
2 import { useParams } from 'react-router-dom';
3 const User = () => {
4   const params = useParams();
5   console.log(params);
6   return (
7     <div>
8       <h1>User : {params.id}회원정보입니다.</h1>
9     </div>
10  );
11 };
12
13 export default User;
```

## 👁️ queryString을 이용한 페이지 이동

The screenshot shows a web application running on localhost:3000/admin?name=heejung. The browser address bar is highlighted with a pink box. The page displays "Admin입니다 heejung" (Admin is heejung). The React DevTools console shows the URLSearchParams object for the query string. The code in the editor shows the useSearchParams hook being used to retrieve the name from the query string.

```
1 import React from 'react';
2 import { useSearchParams } from 'react-router-dom';
3
4 const Admin = () => {
5   const [params, setParams] = useSearchParams();
6   console.log(params);
7   console.log(params.get("name"));
8   return (
9     <div>
10       <h1>Admin입니다 {params.get("name")}</h1>
11     </div>
12   );
13 };
```