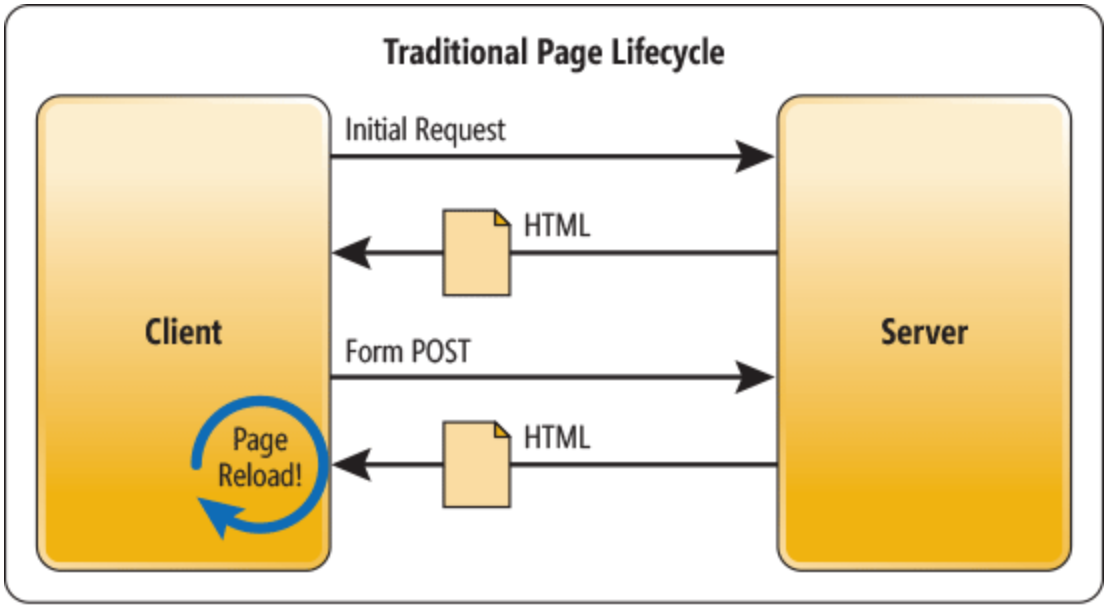
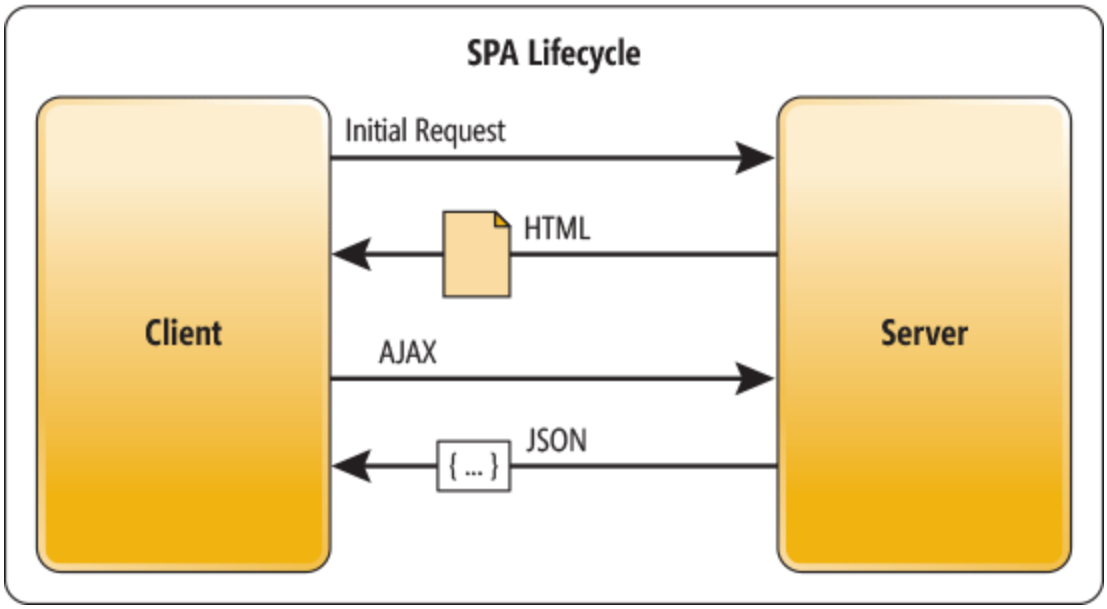
6 React-Router

SPA vs MPA





MPA

- 유저가 **페이지 이동을 할 때마다** html 파일을 받고, 추가적인 리소스를 전달받아 렌더링
- SPA 개념이 생기기 전 기존 방식
- 사용자 인터렉션이 많은 경우 위 과정을 반복했을 때 서버의 트래픽과 리소스가 많이 필요하다.
- **SSR**(Server Side Rendering)
  - **서버에서** 콘텐츠 렌더링을 하는 방식

SPA

html 파일을 한 번만 받아오고 그 이후에는 **필요한 데이터는 받아와서 렌더링**

- 한 페이지만 존재하고 유저의 브라우저 **주소창의 경로에 따라 알맞은 페이지를 보여주는 것**
- 서버에서 html과 추가적인 리소스가 아닌 **필요한 데이터만을 받아와서 업데이트** 해주는 것
- **CSR**(Client Side Rendering)
  - 클라이언트에서 처음 html 파일을 통해 **유저의 브라우저에서** 콘텐츠 렌더링을 하는 방식

단점

- 웹페이지 초기 렌더링 비용이 크다.

  → 시간이 오래걸린다.
- 검색 엔진 최적화?
  - **많은 크롤러들이 JS를 지원하지 않기 때문에** JS로 렌더링하는 SPA는 검색 엔진 최적화에 적합하지 않다 라는 말이 있다.
  - 구글 크롤러는 JS 를 지원하기 때문에 아예 크롤링 할 수 없는 것은 아니지만 **SSR 방식과 비교했을 때 CSR이 검색 엔진 최적화가 덜 된다는 단점**을 가지는 것은 맞다.

React-Router

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

BrowserRouter

- **브라우저의 history API**를 이용해서 새로고침 없이 주소를 변경하고 **주소에 맞는 컴포넌트를 렌더링** 할 수 있게 해준다.

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

Routes/Route

- 주소 경로에 맞는 컴포넌트를 렌더링 하려면 **주소와 컴포넌트를 맵핑**시켜줘야 한다.

```
import { Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import BoardsList from "./pages/BoardsList";
import Board from "./pages/Board";

function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/boards" element={<BoardsList />} />
      <Route path="/boards/:id" element={<Board />} />
    </Routes>
  );
}

export default App;
```

- **Routes** : Route 컴포넌트는 Routes 내부에서 사용되어야 한다.
- **Route** : **경로(path)**와 **렌더링할 컴포넌트(element)**를 지정해준다.
  - ':'을 이용해 **URL Parameter**를 전달할 수 있다. (아래에서 어떻게 다룰지 볼 예정)

Link

- a 태그를 이용했을 때 **새로고침 되는 현상을 막기 위해서 사용**
- **새로고침 없이** 브라우저 주소의 경로만 바꿔준다.

```
import React from "react";
import { Link } from "react-router-dom";

const Home = () => {
  return (
    <>
      <Link to="/boards">Boards</Link>
    </>
  );
};

export default Home;
```

useNavigate

- Link를 이용하면 **눌렀을 때** 새로고침 없이 페이지 이동을 시킬 수 있다.
- **useNavigate**
  - **앞/뒤**로 이동 시킬 수 있다.
  - **어떤 조건을 만족했을 때** 페이지 이동을 시킬 수 있다.
  - **state를 이용**해 값을 전달할 수 있다.

  ex) 클라이언트 상황에 따라 다른 페이지로 이동시켜야 할 수 있다.

```
import { Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import BoardsList from "./pages/BoardsList";
import Board from "./pages/Board";
import { useNavigate } from "react-router-dom";

function App() {
```

```
  const navigate = useNavigate();

  const goBack = () => {
    navigate(-1);
  };

  const goBoards = () => {
    navigate("/boards", {
      state: { message: "여긴 게시판입니다." },
    });
  };

  return (
    <>
      <header style={{ background: "lightgray", padding: 16, fontSize: 24 }}>
        <button onClick={goBack}>뒤로가기</button>
        <button onClick={goBoards}>게시글 목록</button>
      </header>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/boards" element={<BoardsList />} />
        <Route path="/boards/:id" element={<Board />} />
      </Routes>
    </>
  );
}

export default App;
```

URL Parameter/QueryString

URL Parameter

- 주소의 경로에 유동적인 값을 넣는 형태

```
boards/1
```

- 일반적으로 파라미터는 특정 id 같은 **식별자로 조회**할 때 사용

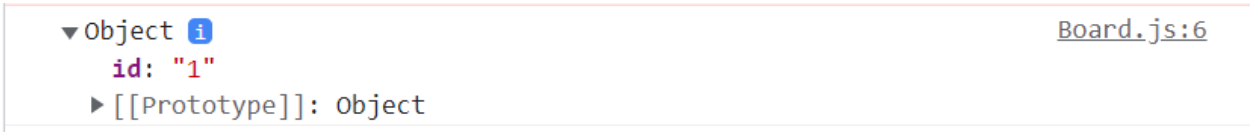
URL Parameter를 어떻게 가져올까?

- **UseParams**

```
import React from "react";
import { useParams } from "react-router-dom";

const Board = () => {
  const params = useParams();
  console.log(params);
  return <div>hi</div>;
};

export default Board;
```



- 객체의 키 값은 **Route 컴포넌트에서 지정해준 변수**로 지정된다.

QueryString

- 주소의 뒷부분에 '?' 문자 이후 **key=value 형태값을 전달**하는 방식

```
boards?detail=true
```

- 키워드를 검색하거나 **요청에 필요한 옵션**을 전달할 때
- **정렬**, **필터링**할 때 사용

QueryString을 어떻게 가져올까?

- **useLocation**

```
import React from "react";
import { useLocation } from "react-router-dom";

const BoardsList = () => {
  const query = useLocation();
  console.log(query);
  return <div>boardsList</div>;
};
```

```
export default BoardsList;
```

- search를 키값으로 **'?'를 포함한 쿼리스트링을 값**으로 가지는 객체
  - qs라는 라이브러리를 이용해 **쿼리스트링을 파싱**할 수 있다.

실습

- 영화 포스터를 눌렀을 때 영화 세부 페이지로 이동
- 세부 정보를 렌더링