2023.03.19 SPA, CSR, SSR

1. SPA(Single Page Application) vs MPA(Multi Page Application)
    a. SPA는 말 그대로, **하나의 페이지로 이루어진 어플리케이션**을 의미하는 모던 웹의 패러다임입니다.
    b. 전통적인 MPA는 **a 태그**를 이용한 방식으로, 페이지 이동 시마다 HTML과 같은 정적 리소스를 서버에 요청하고, 이를 다운로드하여 **전체 페이지를 다시 렌더링**합니다. 따라서 페이지 이동 시마다 새로고침이 발생하고 사용성이 떨어지며, 변하지 않는 부분까지 포함하여 페이지가 갱신되므로 비효율적입니다.
    c. 반면, SPA는 기본적으로 웹 어플리케이션에 필요한 모든 정적 리소스를 **최초 한 번만 다운로드**합니다. 이후 페이지 갱신이 필요할 때마다 **필요한 데이터를 JSON으로 전달**받아 변경되는 부분을 갱신합니다. 변경되는 부분만을 갱신하므로 전체적인 트래픽을 감소시킬 수 있고, 새로고침이 발생하지 않아 네이티브 앱과 유사한 사용자 경험을 제공할 수 있습니다.

> 네이티브 앱이란, 각 모바일 플랫폼에서 제공하는 언어(네이티브 언어)로 만들어진 앱을 의미하며, 해당 플랫폼에 최적화된 언어로 개발되었기 때문에 반응 속도가 빠르고 안정적입니다.

2. CSR(Client Side Rendering) vs SSR(Server Side Rendering)
    a. 각각 SPA, MPA에서 사용하는 렌더링 방식으로, CSR이란 **사용자의 요청에 따라 필요한 부분만 응답 받아 렌더링**하는 것이고, SSR은 **서버로부터 완전하게 만들어진 html 파일을 받아와 페이지 전체를 렌더링**하는 것입니다.
    b. SSR
        i. 브라우저에서 요청을 보내면 서버에서 데이터 및 스타일을 모두 삽입하여 브라우저에 전달합니다.
        ii. 사용자 인터렉션으로 인해 일부 데이터가 변경되면 서버에서 **새롭게 화면을 그려서 브라우저에 전달하므로 인터렉션이 많은 웹 페이지에서는 비효율적**이고 페이지 이동이 느립니다. 이는 서버의 부하를 증가시키기도 합니다.
        iii. 또한, 브라우저가 HTML과 JS를 다운로드 받을 때의 시간 차이 때문에, **TTV(Time To View)와 TTI(Time To Interact) 사이의 시간 차이가 발생**하게 되고, 이로 인해 사용자 인터렉션이 응답하지 않을 수 있습니다.

iv. 반면에, 화면을 구성하는 각각의 페이지가 존재하고, 화면을 구성하는 데이터가 이미 HTML에 담겨진 채로 브라우저에 전달되기 때문에, **검색 엔진 최적화 (Search Engine Optimization, SEO)에는 유리**합니다.

v. 또한, 해당 페이지만을 주고받기 때문에 **초기 로딩 속도가 빠릅니다.**

c. CSR

i. 브라우저에서 요청을 보내면 먼저 빈 html 화면을 보여주고, 화면을 표시하는 데 필요한 모든 JS 파일을 전달합니다.

ii. 초기 화면 이외에도 모든 리소스를 전달하기 때문에, **초기 로딩 속도가 느립니다.**

> Webpack은 기본적으로 모듈 번들러이지만, 코드 스플리팅(spliting)을 지원하여 chunk 단위로 JS 파일을 주고받을 수 있게 해 줍니다. 따라서 CSR 방식에서 초기 로딩 속도 문제를 해결하기 위하여 사용되기도 합니다.

iii. 또한, 하나의 페이지만을 사용하므로 **검색 엔진 최적화에 불리**합니다.

> **PJAX**란, Pushstate + Ajax의 줄임말로, 브라우저의 pushstate, popstate를 이용해 서버에 요청은 보내지 않으면서 페이지마다 고유의 url을 갖도록 하여 history를 관리할 수 있게 하는 방식입니다. React-router에서는 Browser Router에 해당하는 기능이며, SEO 문제를 해결할 수 있습니다. 참고로 Hash Router는 URI에서 #로 시작하는 문자열인 Hash가 변경되어도 서버에 요청을 보내지 않는다는 점을 이용하여 페이지를 구분하도록 하는 **Hash 방식**을 사용합니다. 이는 history 관리가 가능하지만 동일한 URL 내에서 URI만이 변경되는 것이므로 SEO 이슈는 여전히 존재합니다.

iv. 사용자 인증 관리를 Cookie나 Local Storage에서 수행하는 경우에는 **보안에 취약**점이 생길 수 있다.

v. 반면에, 초기 로딩 이후에는 이미 다운로드된 리소스를 이용해 화면이 변경되므로 **인터렉션이 많은 웹 페이지에서 효율적**입니다.

vi. 변경되는 데이터만을 서버에 요청하므로 **서버의 부담도 줄어들고 새로고침도 발생하지 않습니다**.

d. Universal Rendering

i. **최초 페이지는 SSR 방식, 이후로는 CSR 방식**을 이용함으로써 SSR과 CSR의 장점만을 취할 수 있는 방식입니다.

ii. 최초 렌더링 시에는 빈 html이 아닌, 데이터가 포함된 html을 다운로드 받음으로써 SEO 이슈를 해결하고, 이후로는 데이터만을 주고받으며 서버의 부하를 줄이고 사용자 경험을 향상시키는 방식입니다.

1. React.js ⇒ Next.js
2. Vue.js ⇒ Nuxt.js
3. Angular.js ⇒ Angular Universal