Hydration

💡 Hydration

Hydrate는 Server Side 단에서 렌더링 된 정적 페이지와 번들링 된 JS 파일을 클라이언트에게 보낸 뒤, 클라이언트 단에서 HTML 코드와 React인 JS 코드를 서로 매칭 시키는 과정을 의미합니다.

💡 Hydrate 과정이 필요한 이유

리액트를 사용하다 보면 JS 만을 사용해 웹 화면을 구성한다는 것을 알고 있습니다. 그래서 실제로 HTML 파일에는 안에 내용이 하나도 없습니다.

Next.js의 경우 기본적으로 SSR이 기본이기 때문에, 클라이언트에게 웹 페이지를 보내기 전에 Server Side에서 미리 웹 페이지를 Pre-Rendering 합니다. 그리고 Pre-rendering으로 인해 생성된 HTML document를 클라이언트에게 전송합니다.

그리고 이 시점에서 중요한 내용이 나옵니다.

현재 클라이언트가 받은 웹 페이지는 단순히 웹 화면만 보여주는 HTML일 뿐이고, 자바스크립트 요소들이 하나도 없는 상태입니다. 이는 웹 화면을 보여주고 있지만, 특정 JS 모듈뿐 아니라 단순 클릭과 같은 이벤트 리스터들도 하나도 적용되어 있지 않다는 것입니다.

Next.js 서버는 Pre-rendering 된 웹 페이지를 클라이언트에게 보내고 나서, 바로 리액트가 번들링 된 자바스크립트 코드들을 클라이언트에게 전송합니다.

네트워크 탭을 보면, 맨 처음 응답받는 요소가 document Type의 파일이고, 이후에 React 코드들이 렌더링 된 JS 파일들이 Chunk 단위로 다운로드 됩니다. 그리고 이 자바스크립트

코드들이 이전에 보내진 HTML DOM 요소 위에서 한번 더 렌더링을 하면서, 각자 자기 자리를 찾아가며 매칭이 되는데 이 과정을 Hydrate 라고 합니다.

마치 자바스크립트 코드들이 DOM 요소 위에 물을 채우듯 필요로 하던 요소들을 채운다 하여 Hydrate(수화)라는 용어를 쓴다고 합니다.

> 💡 Server에서 한번 렌더링 하고 Client에서 또 렌더링 하는게 비효율적이지 않은가

물론 비효율적으로 보일 수 있습니다. 하지만 서버 단에서 빠르게 Pre-rendering하고 유저에게 빠른 웹 페이지로 응답할 수 있다는 것은 너무나 큰 이점입니다. 심지어 JS 요소들이 빠진 상태이니 굉장히 가벼워 클라이언트에서 빠른 로딩도 가능하죠. 이는 두 번 렌더링이 된다는 단점을 보완하고도 남습니다.

더 나아가서 클라이언트 단에서 자바스크립트가 렌더링을 할 때, 단지 각 DOM 요소에 자바스크립트 속성을 매칭 시키기 위한 목적이므로 실제 웹 페이지를 다시 그리는 과정까지는 하지 않습니다. (Paint 함수 호출 X)