MPA(multi page application) – 여러 페이지로 구성된 웹 어플리케이션. 사용자의 클릭과 같이 인터렉션이 발생할 때, 서버로부터 새로운 html을 받아와서 해당 링크로 이동하여 페이지 전체를 새로 렌더링하는 웹 페이지 구성 방식.

SPA(single page application) – 하나의 페이지로 구성된 웹 어플리케이션. 브라우저에 최초에 한 번 페이지 전체를 로드하고, 이후부터는 특정 부분만 Ajax를 통해 데이터를 바인딩하는 방식. Ex) react와 vue, 앵귤러와 같은 자바스크립트 프레임워크.

SSR(서버사이드 렌더링)

서버로부터 완전하게 만들어진 html파일을 받아와 페이지 전체를 렌더링 하는 방식.

MPA에서 ssr방식을 채택.

각각의 페이지가 있기 때문에 Seo(검색엔진최적화)에 유리.

서버로부터 화면을 렌더링 하기 위한 필수적인 요소를 먼저 가져오기 때문에 보다 빠른 초기 로딩.

TTV(Time to View)와 TTI(Time to Interact) 간에 시간 간격이 존재. -> 사용자가 버튼을 클릭하거나 이동하려고 해도 아무런 반응이 없을 수 있다.

매번 페이지를 요청할 때마다 새로 고침 되기 때문에 사용자 경험이 다소 떨어짐. (= 화면이 깜빡임)

페이지를 요청할 때마다 서버에서 페이지를 구성하는 모든 리소스를 준비해서 응답해서 서버측 부하가 증가.

1. 클라이언트가 서버에 초기 화면 로드 요청
2. 서버는 화면에 필요한 데이터를 얻어와 모두 삽입하고 css까지 모두 적용해서 렌더링 준비를 마친 HTML과 JS코드를 브라우저에 응답으로 전달
3. 브라우저에서 전달받은 페이지 띄우고, JS코드 다운로드하고 html에서 실행
4. 일부만 변경해도 1,2,3의 과정을 모두 거친다.

CSR(클라이언트 사이드 렌더링)

SPA에서 csr 방식을 채택.

사용자의 요청에 따라 필요한 부분만 응답 받아 렌더링 하는 방식.

Ssr과 다른 점은 모든 js 파일 다운. -> 초기 로딩 시간이 보다 오래 걸린다.

변경된 부분과 관련된 데이터만 가져오기에 보다 빠른 속도.

변경된 부분만 요청함으로 서버 부하 감소.

사용자 친화적. (깜빡임 없는 부드러운 이동)

자바스크립트를 사용하여 사용자와 상호작용 후에 페이지 내용 로드해서 Seo에 불리.

1. 클라이언트에서 서버에 초기화면 로드 요청
2. 서버가 화면에 표시하는데 필요한 완전한 리소스의 응답
3. 일부 변경시 서버는 변경된 부분과 관련된 리소스만 응답.

10분 테코톡 CSR, SSR

SPA, MPA !== CSR, SSR 서로 다른 개념이다.

CSR ClientSideRendering: 클라이언트 측에서 rendering을 준비. User가 Browser에 방문하면 Browser가 Server에 콘텐츠를 요청한다. Server는 빈 뼈대 Html(연결된 js 링크)를 Browser에 보내주고, Browser에서는 js링크를 통해 js를 다운로드 받고, 동적 DOM을 만들어서 Browser에 띄워 User가 볼 수 있다. 동적 DOM 생성으로 초기 로딩 속도가 느리다. 하지만 이후 페이지 일부 변경 시 구동 속도는 빠르다. 빈 뼈대 Html만 보내기 때문에 서버 부하가 적고, client 측에서 연산, routing을 모두 처리하기 때문에 반응 속도가 빠르고 UX(사용자 경험)가 우수하다. 사용자와 상호작용이 많고, 개인 정보를 기준으로 이루어진 페이지들 같이 정보 보호가 더 중요한 경우

텍스트이(가) 표시된 사진

자동 생성된 설명 빈 뼈대 Html

텅텅 비어 있어 검색엔진이 색인 할 만한 컨텐츠가 존재하지 않는다. 따라서 검색엔진 최적화에 불리하다.

초기 로딩 속도 보안: code splitting, tree-shaking, chunk 분리

Seo 개선: pre-rendering

CSR + SSR/SSG: 사용자에 따라 내용이 달라지고, 빠른 인터랙션이 중요하고, 검색엔진 최적화도 필요한 경우

프레임워크 없이. express js로 별도의 서버를 직접 운영. Nest js도 사용가능. 단 서버 환경 구성이나 build 등의 작업이 생소할 경우 진입 장벽이 존재함.

프레임워크 포함. Next js, Gatsby, NUXT, Angular Universal. CSR에 비해 코드 복잡도가 올라가고, 블랙박스 영역이 생긴다.

Isomorphic App, Universal Rendering: 서버와 클라이언트에서 동일한 코드가 동작하는 application. 같은 코드로 돌아가기 때문에 예상과 다른 결과를 맞이할 수 있지만, 초기 로딩 속도 보완, seo 개선, CSR의 장점 유지 등의 이유로 21년 사용되는 추세를 보인다.

SSR ServerSideRendering: 서버 측에서 rendering을 준비. 요청 즉시 만들어 데이터가 달라져서 미리 만들어두기 어려운 페이지에 적합. User가 Browser에 방문하면 Browser가 Server에 콘텐츠를 요청한다. Server에서는 페이지에 필요한 데이터를 가져와 모두 삽입하고 CSS까지 적용해서 렌더링 준비를 마친 Html과 js를 Browser에 전달한다. Browser에서는 전달받은 페이지를 띄우고, js 코드를 다운로드하고 Html에 js 로직을 연결한다. 모든 데이터가 Html에 담겨 전달되기 때문에 검색엔진 최적화에 유리하다. Js 코드를 다운로드 받고 실행하기 전에 User가 화면을 볼 수 있다. 초기 구동 속도가 빠르다. 단점 TTV !== TTI 즉, 버튼을 눌러도 js 코드를 다운받고 로직이 연결되기 전까지는 반응이 없을 수 있다. 회사 홈페이지이기에 상위에 노출되어야 하고, 누구에게나 동일한 데이터를 보여주고 자주 업데이트할 경우

테이블이(가) 표시된 사진

자동 생성된 설명

SSG StaticSiteGeneration: Static Rendering. 서버에 미리 만들어 두어 바뀔 일이 거의 없는 페이지에 적합. 회사 홈페이지이기에 상위에 노출되어야 하고, 누구에게나 동일한 데이터를 보여주고 자주 업데이트하지 않을 경우