CSR vs. SSR

페이지 구성 방식

Single Page Application

원하는 부분만 교체 가능

화면 깜빡임 X

Muti Page Application

페이지 전체만 교체 가능

화면 깜빡임 O

렌더링 방식

Client Side Rendering

초기 로딩 속도 느림

일부 변경 시 구동 속도 빠름

서버 부하 낮음

SEO 불리

Server Side Rendering

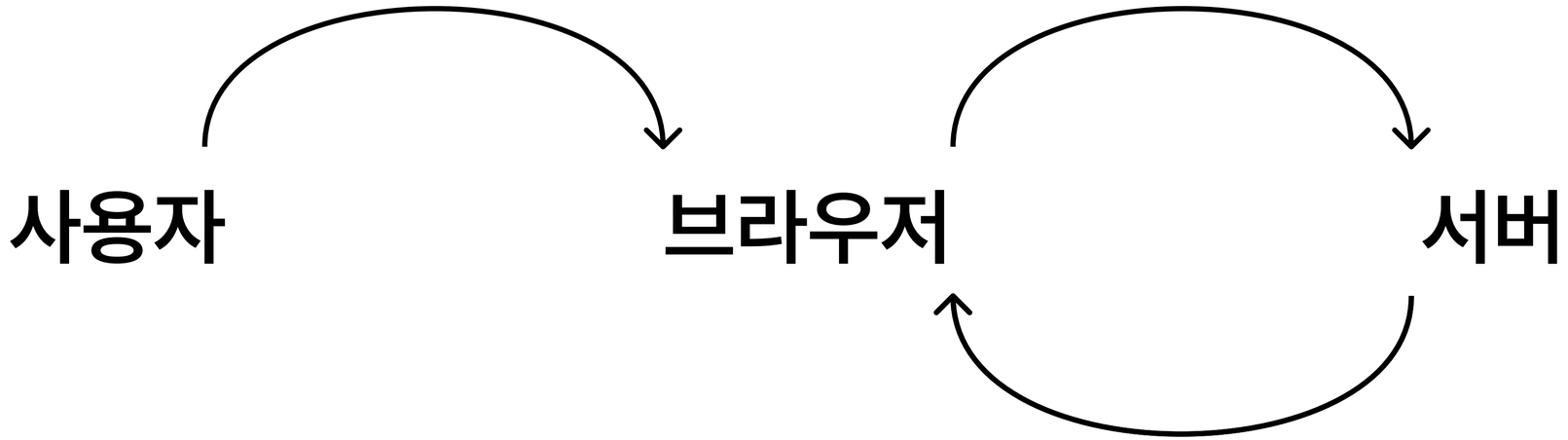
초기 로딩 속도 빠름

Time To View !== Time To Interact

클릭하거나 이동하려해도 무반응일 수 있음

SEO 유리

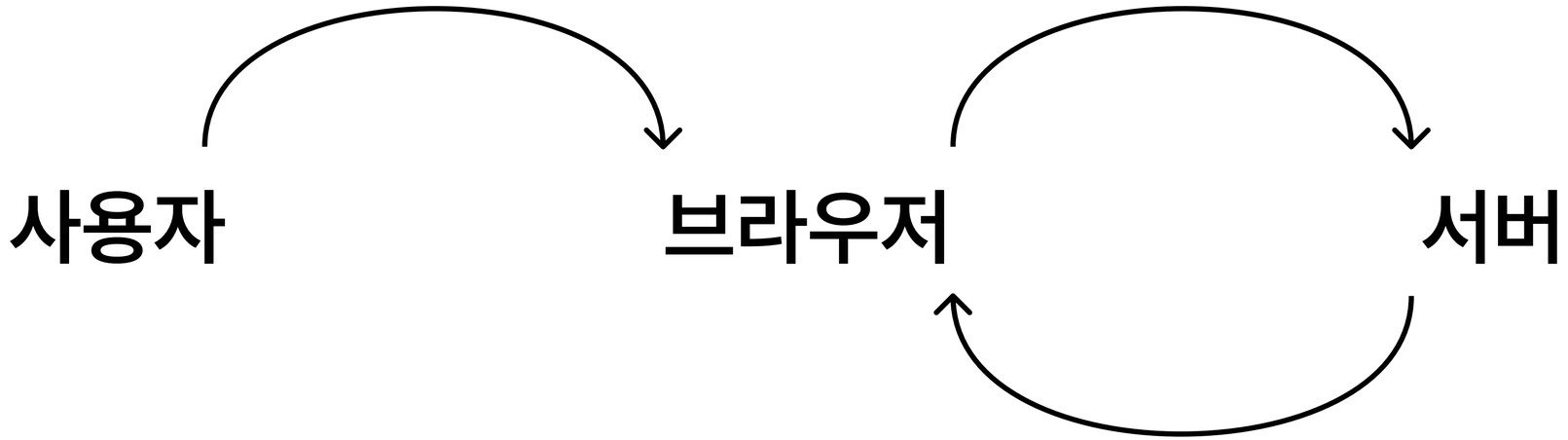
CSR (Client Side Rendering)



JS 링크로 다운

동적 DOM 생성

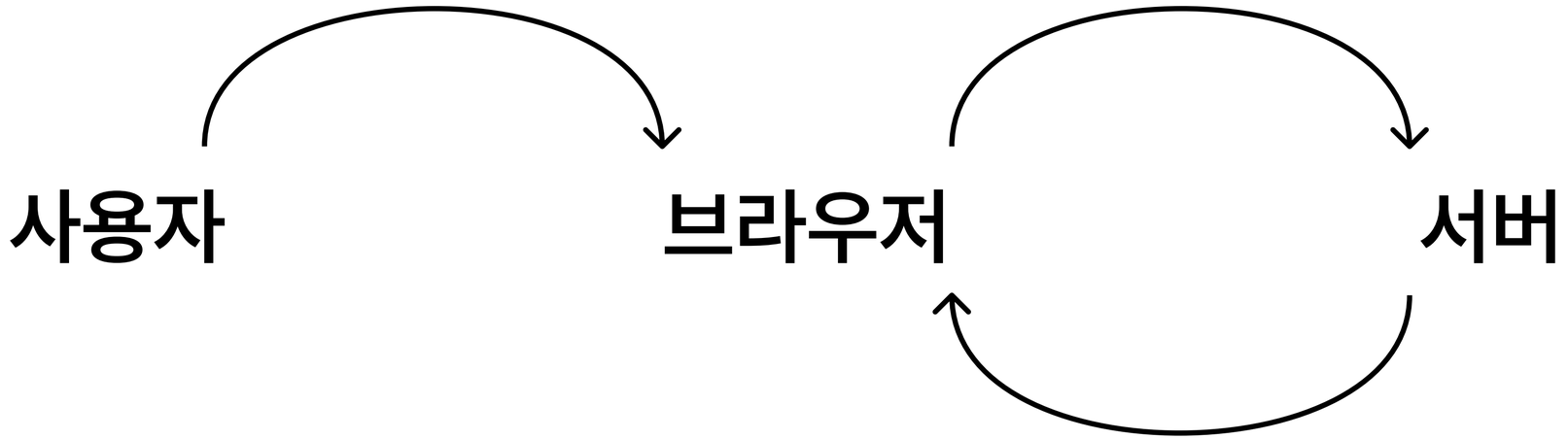
SSR (Server Side Rendering)



바로 전달 받은 페이지 띄움

추가 동적 기능 : JS를 통해 서버에 요청, 받아온 페이지 최신화

SSG (Statice Site Generation, Static Rendering)



페이지에 필요한 HTML, JS, CSS 데이터

페이지를 서버에 모두 만들어 두고 요청 시 해당 페이지 응답

바뀔 일이 거의 없어 캐싱해두면 좋은 페이지에 적합

CSR 보완

초기 로딩 속도 보완

code splitting

tree-shaking

chunk 분리

SEO 개선

pre-rendering

CSR + SSR/SSG

프레임워크 X

express.js로 별도의 서버 직접 운영

nest.js (typescript 기본 지원)

프레임워크 O

react + next.js 페이지별로 SSR, SSG 선택

react + Gatsby SSG에 최적화

vue + nuxt.js

angular(4) + universal

⇒ 코드 복잡도 상승, 제어할 수 없는 블랙박스 영역 존재

CSR + SSR = Isomorphic App, Universal Rendering

초기 로딩 속도 보완

SEO 개선

클라이언트와 서버 모두 같은 코드로 동작하여
예상과 다른 결과있을 수 있음

어떤 렌더링 방법을 선택해야 할까?

렌더링 방식	SEO	상호작용	페이지 데이터 업데이트	사용자 경험 중요	e.g.
CSR	X	많음	.	O	고객 개인 정보 중심
SSR	O	적음	매주 업데이트 O	X	빠른 초기 로딩 SEO가 중요한 서비스
SSG	O	적음	매주 업데이트 X	X	정적인 콘텐츠 중심
CSR + SSR	O	많음	.	O	동적인 상호작용과 SEO가 모두 중요한 서비스에 적합