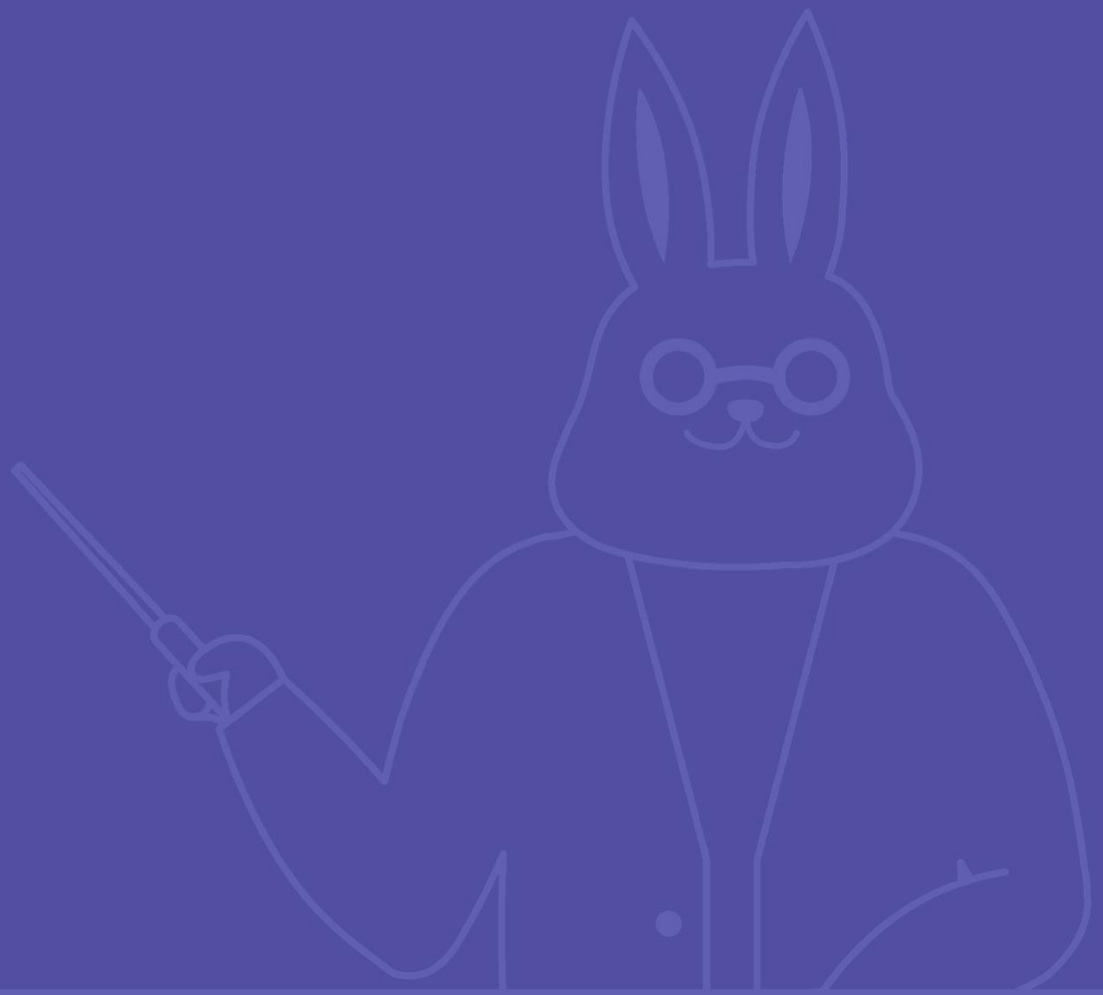
React 심화 II

03 Server Side Rendering

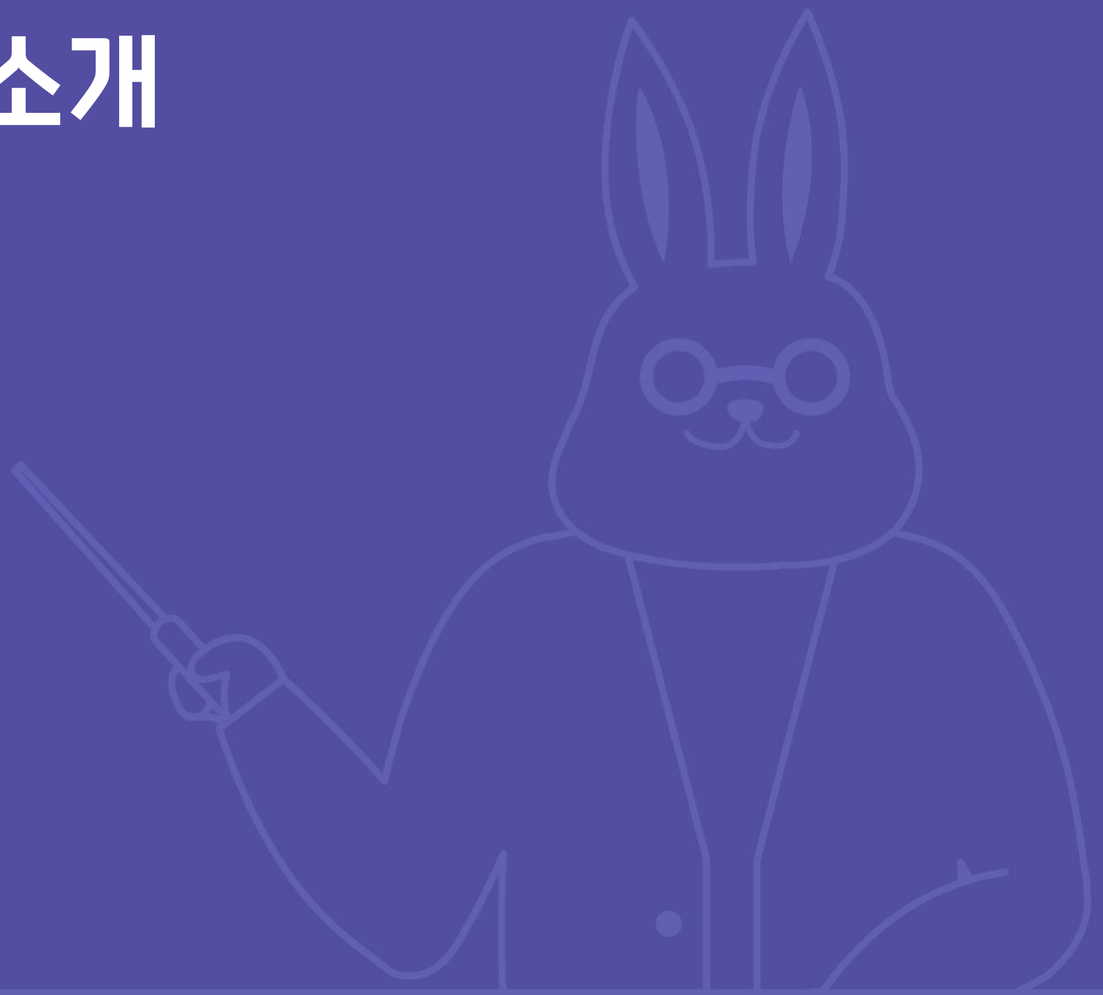


/* elice */

목차

01

Server Side Rendering 소개



Server Rendering

- React, Vue, Angular 등 자바스크립트 프레임워크가 나오기 이전 초기 웹 환경에서는 모든 페이지를 서버에서 빌드.
- 클라이언트는 별도의 처리 없이 웹페이지 노출.
- 이를 Server Rendering이라고 함.

Client Side Rendering

- Ajax 등의 기술, 자바스크립트 프레임워크를 활용하여,
  데이터를 받아 자바스크립트로 페이지를 동적으로 만들 수 있게 됨.
- 데이터는 XML, JSON 형태로 클라이언트에 전송.
- 이를 CSR(Client Side Rendering)이라고 함.

CSR의 장점

- CSR은 자바스크립트만으로 완전히 페이지를 만들 수 있음.
- 자바스크립트를 최대한도로 활용하여 HTML, CSS를 동적으로 생성.
- 컴포넌트 단위로 코드를 나누고, 다양한 디자인 패턴을 적용하는 등, 클라이언트 개발의 수준을 한 단계 끌어올림.
- Full page load 없이 라우팅.

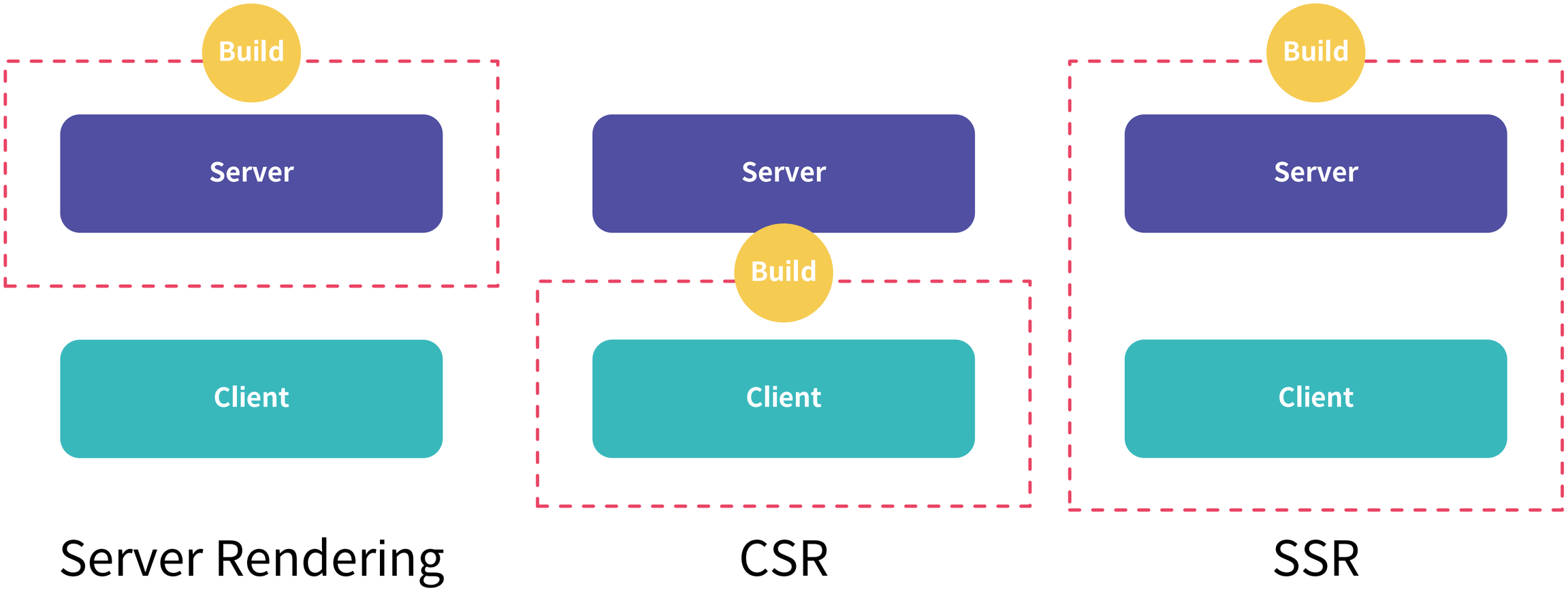
CSR의 단점

- 자바스크립트 코드가 많으면 앱 로딩이 느려짐.
- SEO가 좋지 않음.

Server Side Rendering

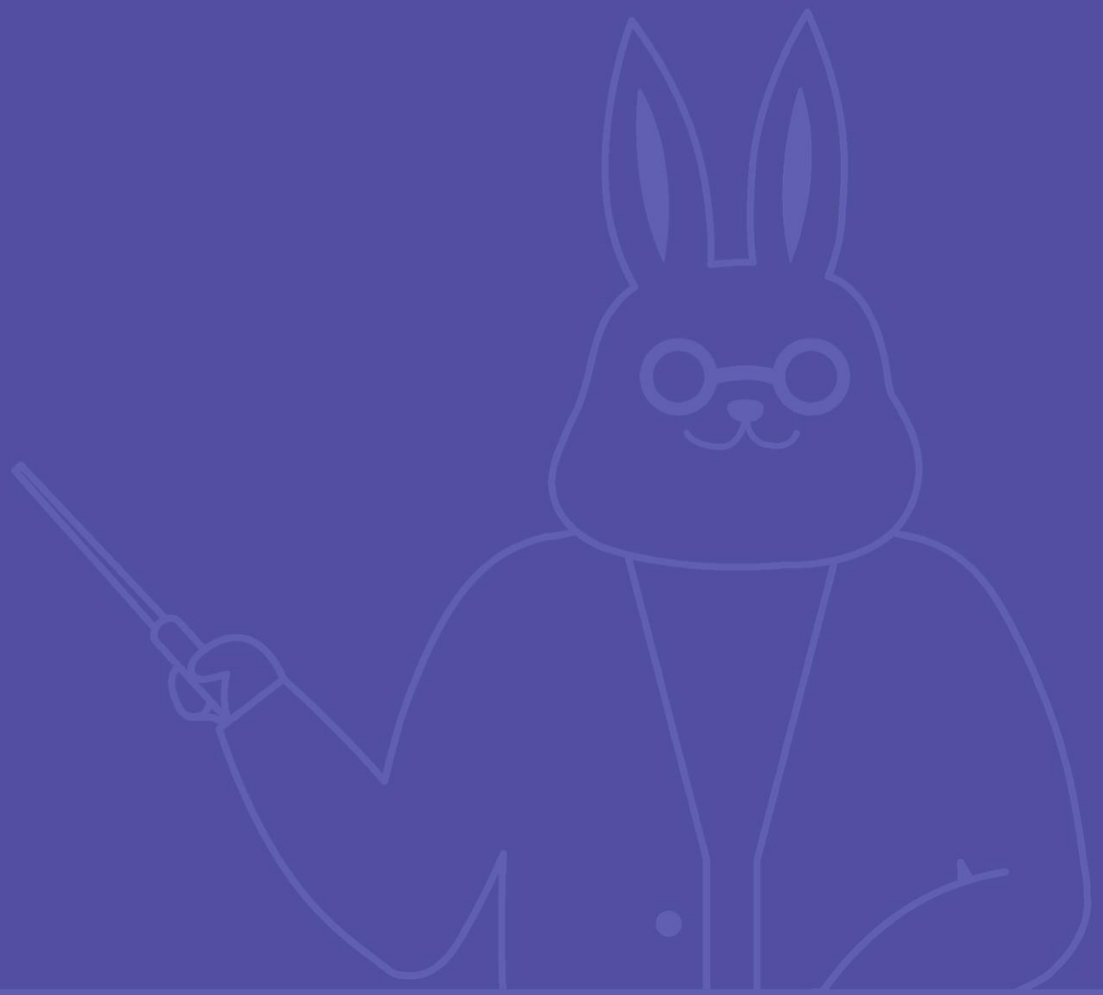
- 서버에서 자바스크립트를 이용해 페이지를 미리 빌드.
- 컴포넌트 생성에 필요한 API 요청, routing, redux store 생성 등을 처리.
- 클라이언트는 빌드된 페이지와 자바스크립트를 받아,
  웹앱을 CSR처럼 동작하게 함.
- 이런 특징으로, Universal Rendering이라고도 함.

Server Side Rendering



02

성능 측정 키 메트릭



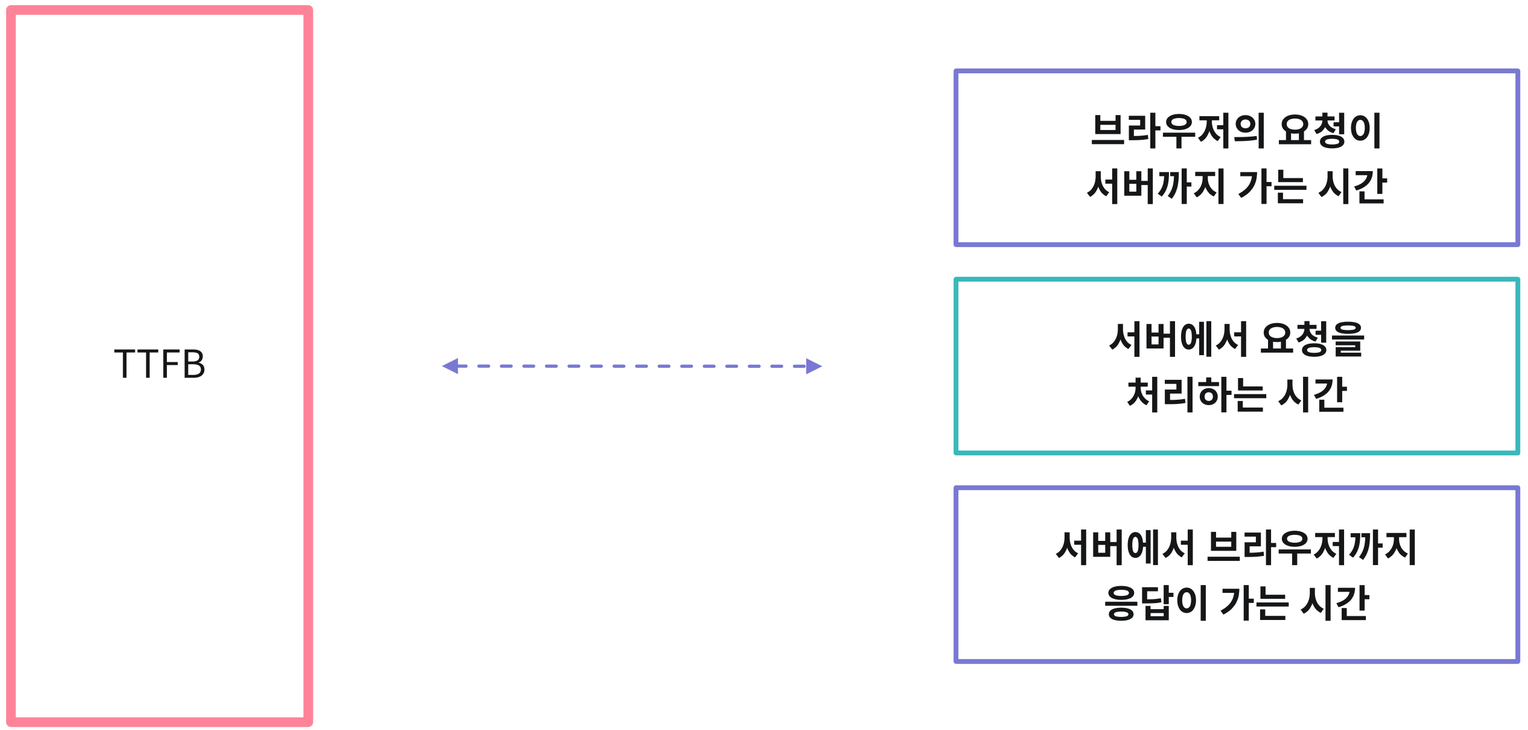
웹 퍼포먼스

- 웹 페이지가 로드되고 유저와 상호작용하는 모든 것들을 측정.
- 성능을 측정하여 웹앱의 사용성을 개선할 수 있음.
- 열악한 네트워크 환경에서도 사용 가능한 앱을 만드는 등
  좋은 유저 경험으로 유저의 만족을 얻음.

Time To First Byte

- 페이지 요청 후, 처음 데이터가 도착하기까지 걸리는 시간.
- 요청을 받았을 때, 서버에서 처리하는 시간이 오래 걸리거나,
  네트워크가 딜레이되는 등의 상황 발생 시 지표가 악화됨.

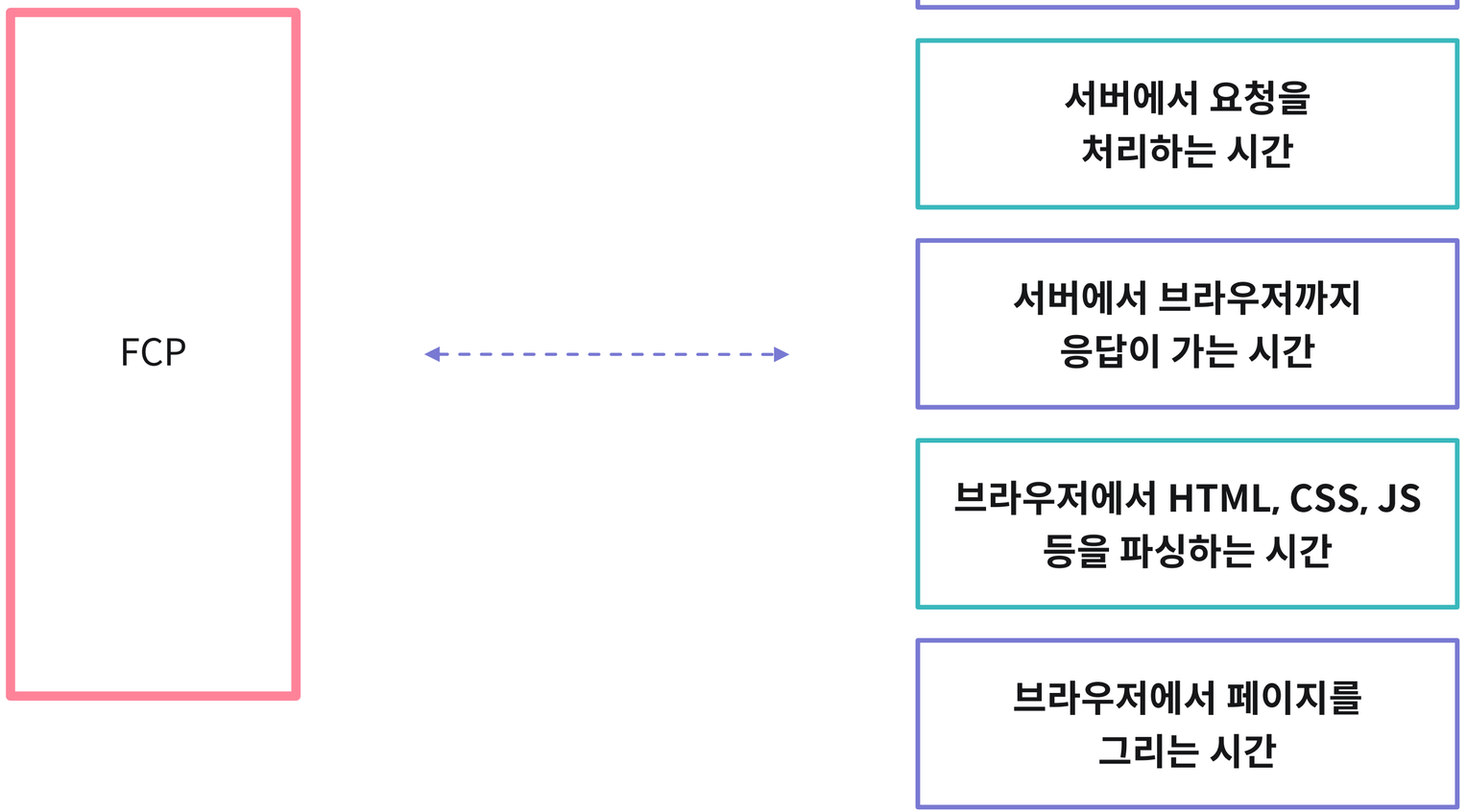
Time To First Byte



First Contentful Paint

- 페이지에 진입하고부터, 브라우저가 어떤 DOM Content를 만들 때까지 걸리는 시간.
- 페이지 진입 후 FCP까지 평균 3초 이상 걸리면 성능 개선이 필요.

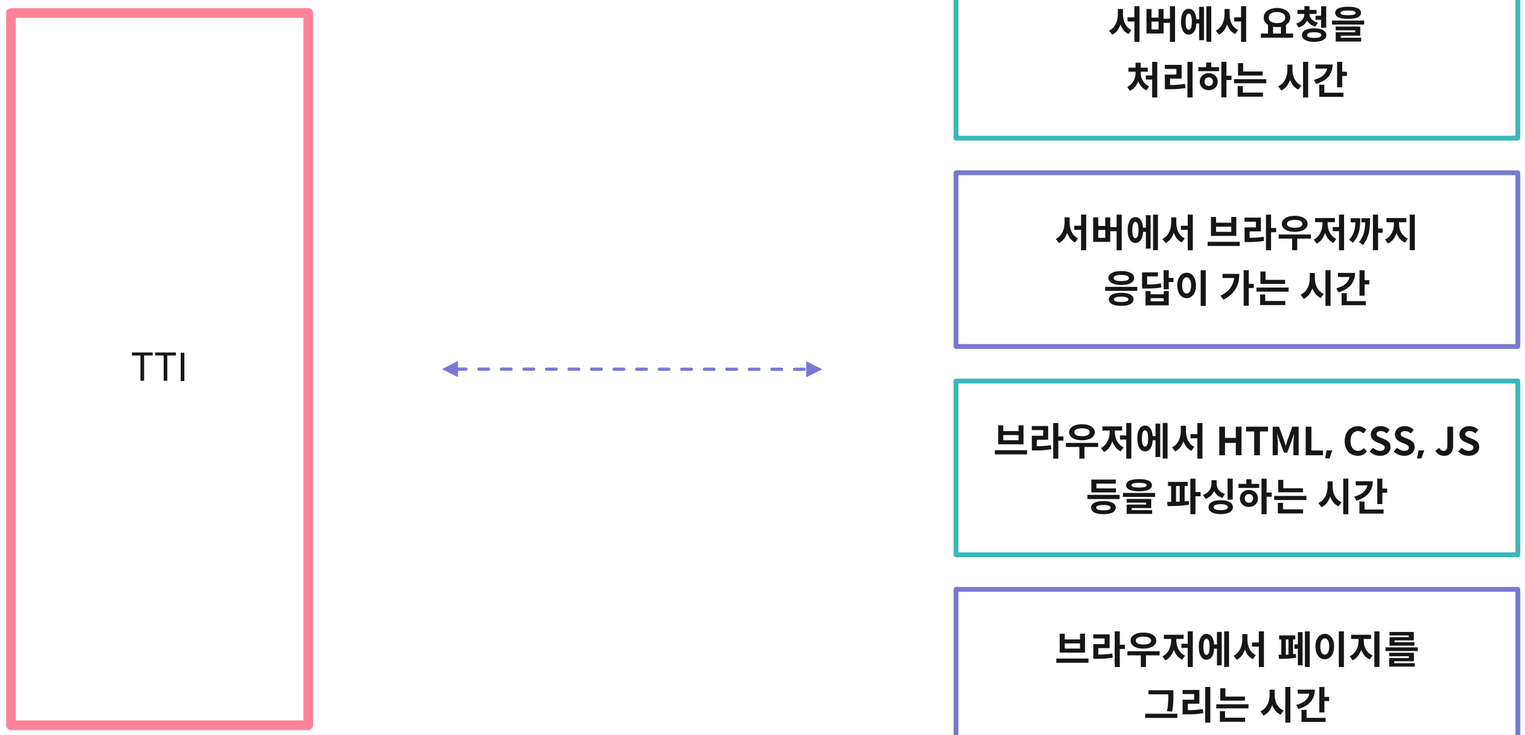
First Contentful Paint



Time to Interactive

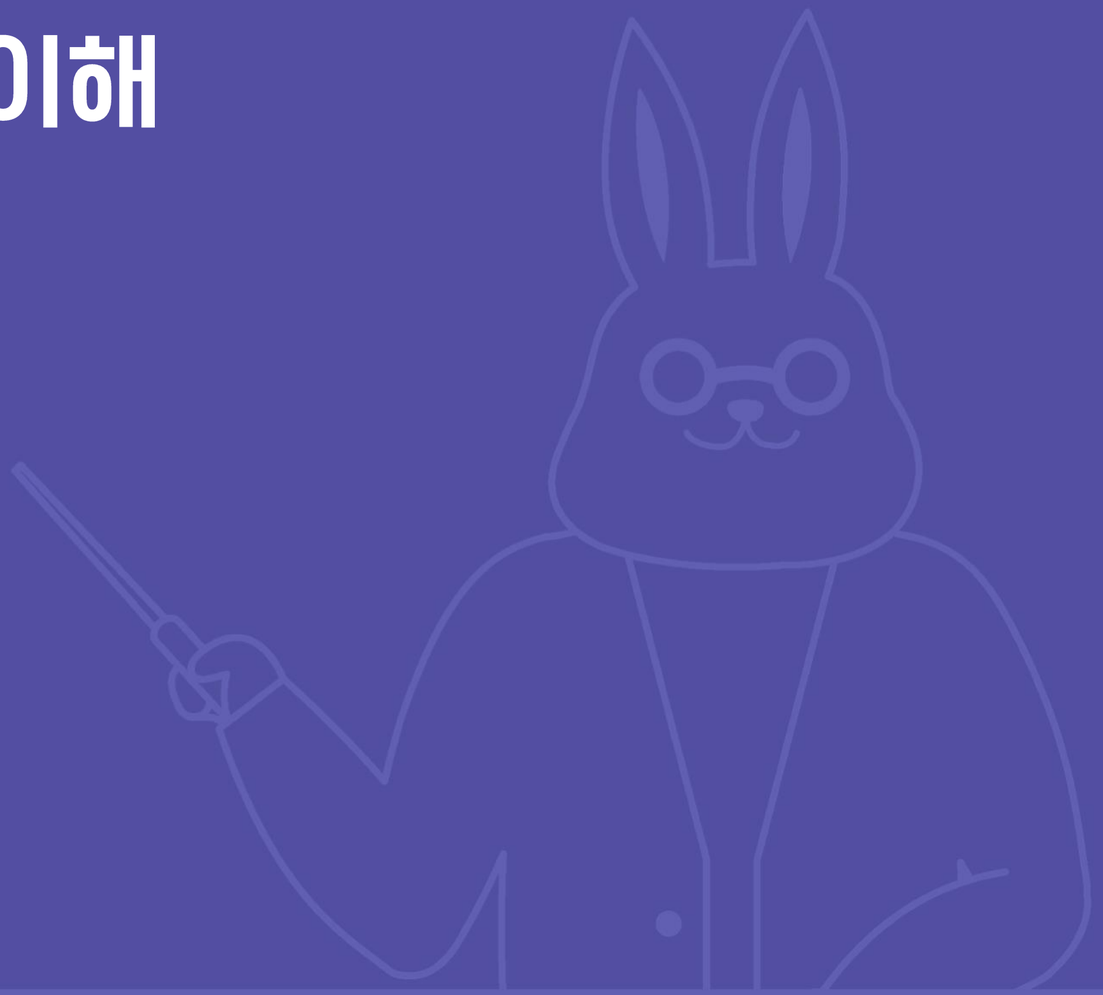
- 웹페이지 진입 후, 유저가 클릭, 스크릭, 인풋 등의 행위를 하기까지 걸리는 시간.
- 자바스크립트가 로드되고 나서, 이벤트 핸들러 등이 부착되어
  입력을 처리할 수 있기까지의 시간.

Time to Interactive

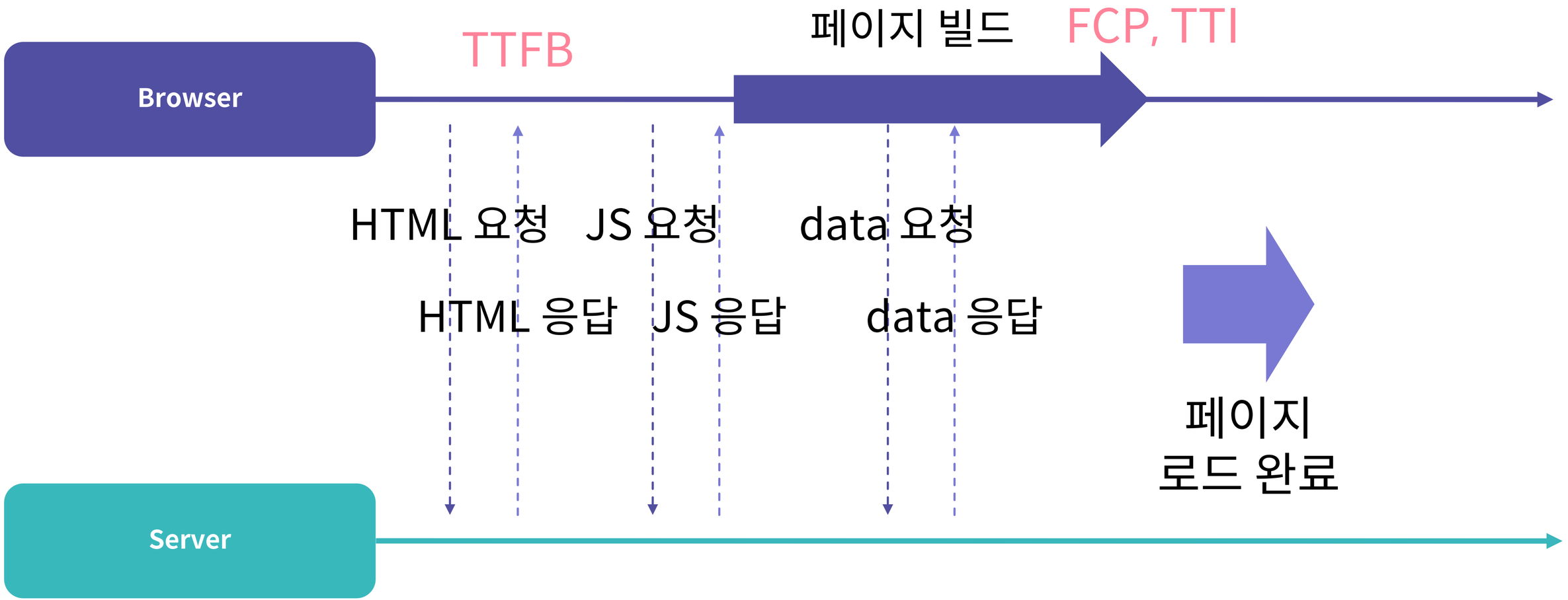


03

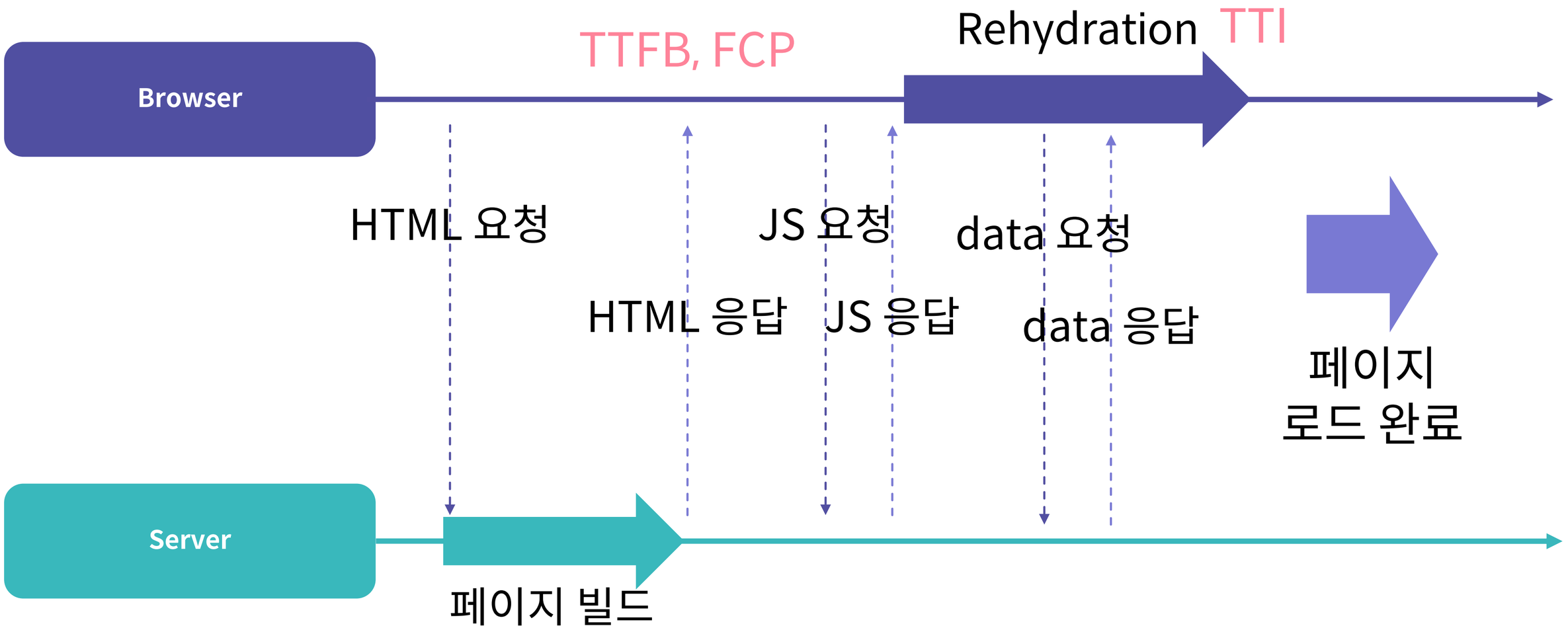
Server Side Rendering 이해



CSR의 페이지 로드 방식



SSR의 페이지 로드 방식



SSR의 페이지 로드 방식

- 유저가 빠르게 페이지의 내용을 볼 수 있도록 HTML을 미리 빌드하여 FCP 등의 키 메트릭을 개선함.
- 서버 자원을 활용하여, 초기 큰 성능이 필요한 페이지 등을 빌드하는 데 활용.

SSR의 장점

- Crawler는 페이지를 Indexing 하기 위해 페이지에 관한 많은 정보가 필요.
- SSR을 활용하여 미리 페이지를 빌드하면, Crawler에게 많은 정보를 줄 수 있음.
- SEO(Search Engine Optimization)에 유리.

SSR의 단점

- CSR에 비해 TTFB에 불리함.
- 별도의 서버를 유지하는 비용.
- Static rendering보다 CDN Caching에 불리.

04

React를 활용한 Server Side Rendering

ReactDOMServer

- ReactDOMServer를 활용하여, 특정 React Component를 HTML로 빌드.
- Node.js 서버에서 JSX를 사용하여 페이지 빌드.

renderToString

- React Component를 HTML로 변환함.
- 클라이언트의 페이지 요청 시, 변환된 HTML string을 전달.
- renderToNodeStream은 readable stream을 생성.
  브라우저가 받아서 점진적으로 페이지를 그림.

ReactDOM.hydrate

- renderToString으로 생성한 HTML의 root을 기준으로,
  받아온 React code를 통해 markup에 이벤트 핸들러를 등록하는 등
  컴포넌트화.

Hydration 시 주의할 점

- 서버에서 생성한 컴포넌트와 브라우저에서 Hydration을 거친 후의 마크업이 다르면, React runtime은 경고를 보냄.
  ex) 현재 시간을 보여주는 컴포넌트
- 경고 발생 시, 어느 부분에서 차이점이 생기는지 반드시 파악해야 함.

Hydration 시 주의할 점

- componentDidMount 역할을 하는 useEffect의 경우, SSR 시 서버에서 동작하지 않음.
- data loading 등의 처리를 별도로 해주어야 할 필요가 있음.

크레딧

/* elice */

코스 매니저
이재성

콘텐츠 제작자
김일식

강사
김일식

감수자
-

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io