


PRE-ONBOARDING CHALLENGE

잠시만 기다려 주시기 바랍니다.





Next.js

확실히 알고 레벨업 하기

Week 1-2

React Server Component 1

1. 과정 오리엔테이션

강의 진행 미리보기

Week 1-1

- 강의 OT
- Next.js의 특징
- 13+ 버전의 기능
- 프로젝트 세팅

Week 1-2

- React 18 동시성
- RSC 란
- RSC 이점

Week 2-1

- RSC의 동작원리

Week 2-2

- 상황에 맞게 코드 설계
- 가장 많이 하는 실수
- 마치며



목차

2시간 미리보기

1. React 18 의 동시성
2. React Server Component
3. React Server Component의 이점

1. React 18 의 동시성

1. React 18 의 동시성

React 18

React v18.0

March 29, 2022 by [The React Team](#)

React 18 is now available on npm! In our last post, we shared step-by-step instructions for [upgrading your app to React 18](#). In this post, we'll give an overview of what's new in React 18, and what it means for the future.

Our latest major version includes out-of-the-box improvements like automatic batching, new APIs like `startTransition`, and streaming server-side rendering with support for `Suspense`.

Many of the features in React 18 are built on top of our new `concurrent` renderer, a behind-the-scenes change that unlocks powerful new capabilities. `Concurrent` React is opt-in — it's only enabled when you use a `concurrent` feature — but we think it will have a big impact on the way people build applications.

We've spent years researching and developing support for concurrency in React, and we've taken extra care to provide a gradual adoption path for existing users. Last summer, [we formed the React 18 Working Group](#) to gather feedback from experts in the community and ensure a smooth upgrade experience for the entire React ecosystem.

In case you missed it, we shared a lot of this vision at React Conf 2021:

- In [the keynote](#), we explain how React 18 fits into our mission to make it easy for developers to build great user experiences
- [Shruti Kapoor demonstrated how to use the new features in React 18](#)
- [Shaundai Person](#) gave us an overview of [streaming server rendering with Suspense](#)

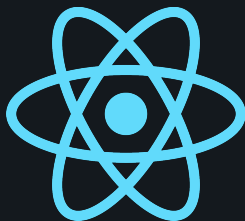
동시성 (Concurrency) 을 제공하기 위해 노력함

1. React 18 의 동시성

갑자기 React 18 ?

NEXT .JS

Framework



1. React 18 의 동시성

갑자기 React 18 ?



v.18 - React Server Component



v.13 - React Server Component 지원

1. React 18 의 동시성

CSR 동작 방식

1. JS 파일 로드

2. 데이터 fetching

3. 컴포넌트 렌더링

4. 사용자 상호작용

1. React 18 의 동시성

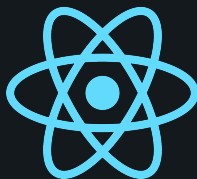
CSR 동작 방식



html / css, js



server api 요청



데이터 포함해 렌더링

1. JS 파일 로드

2. 데이터 fetching

3. 컴포넌트 렌더링

4. 사용자 상호작용

1. React 18 의 동시성

CSR 동작 방식

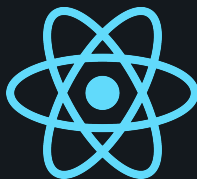
흰색 화면만 보이는 구간



html / css, js



server api 요청



데이터 포함해 렌더링

1. JS 파일 로드

2. 데이터 fetching

3. 컴포넌트 렌더링

4. 사용자 상호작용

1. React 18 의 동시성

CSR 동작 방식

흰색 화면이 오래 보여짐

- 용량이 큰 js
- 네트워크 성능 낮음
- 기기 성능 낮음



html / css, js

1. JS 파일 로드



server api 요청

2. 데이터 fetching



데이터 포함해 렌더링

3. 컴포넌트 렌더링

4. 사용자 상호작용

1. React 18 의 동시성

SSR 동작 방식

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

1. React 18 의 동시성

SSR 동작 방식



server api 요청



.html 파일 전달



js



handler 등록

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

1. React 18 의 동시성

SSR 동작 방식



1. React 18 의 동시성

CSR - SSR

CSR

1. JS 파일 로드

2. 데이터 fetching

3. 컴포넌트 렌더링

4. 사용자 상호작용

상호작용 할 수 없는 html 을 보다 빨리 보여주어 UX 개선

SSR

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration



1. React 18 의 동시성

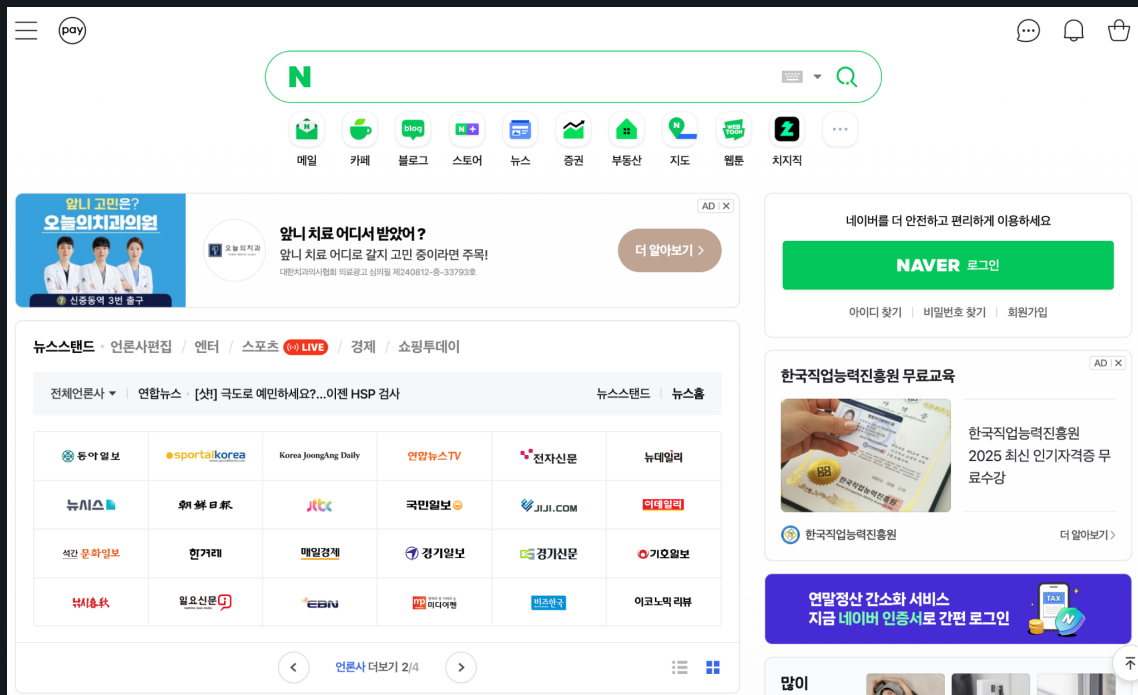
React 18 이전 버전의 문제점

모든 Fetching이 완료되기 전까지는

어떤것도 보여줄 수 없다.

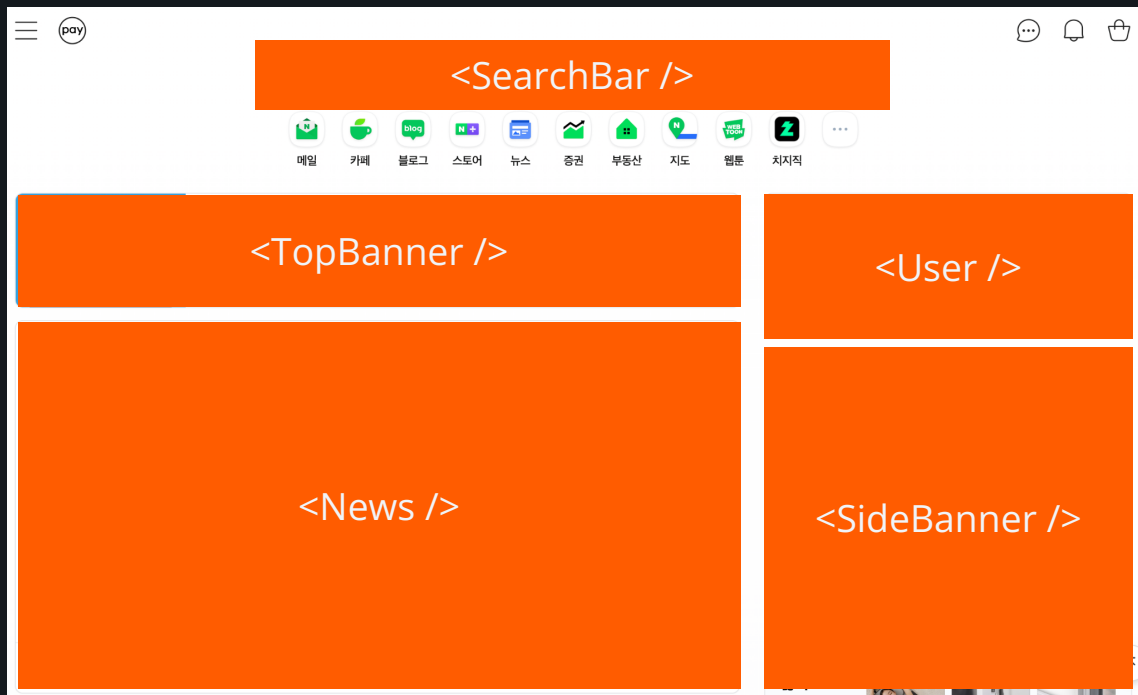
1. React 18 의 동시성

React 18 이전 버전의 문제점



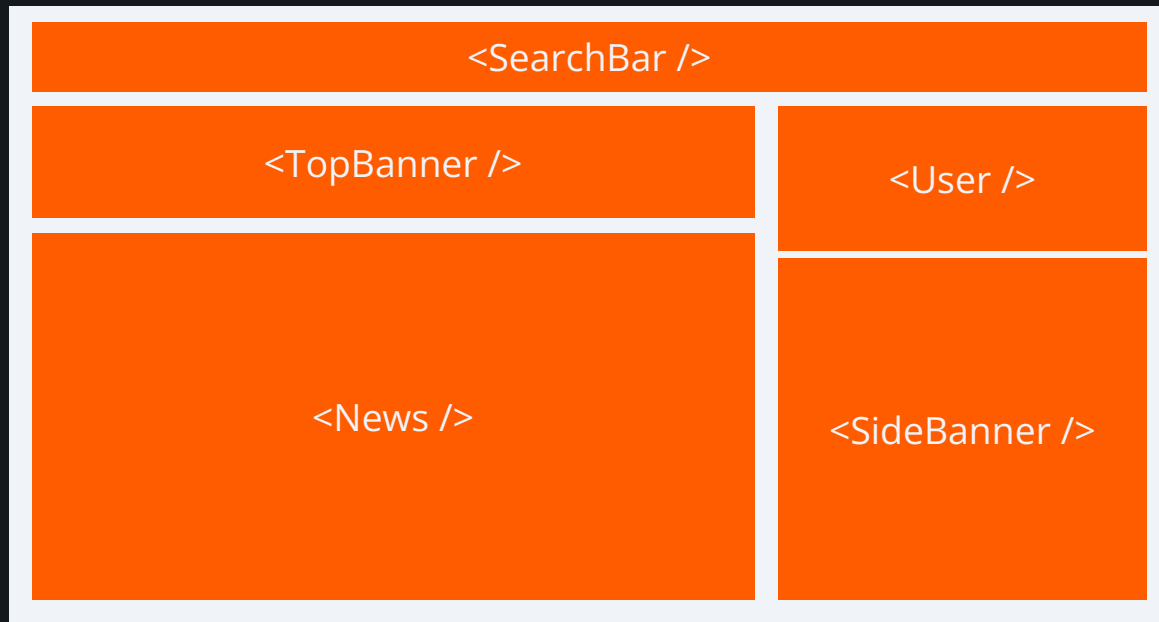
1. React 18 의 동시성

React 18 이전 버전의 문제점



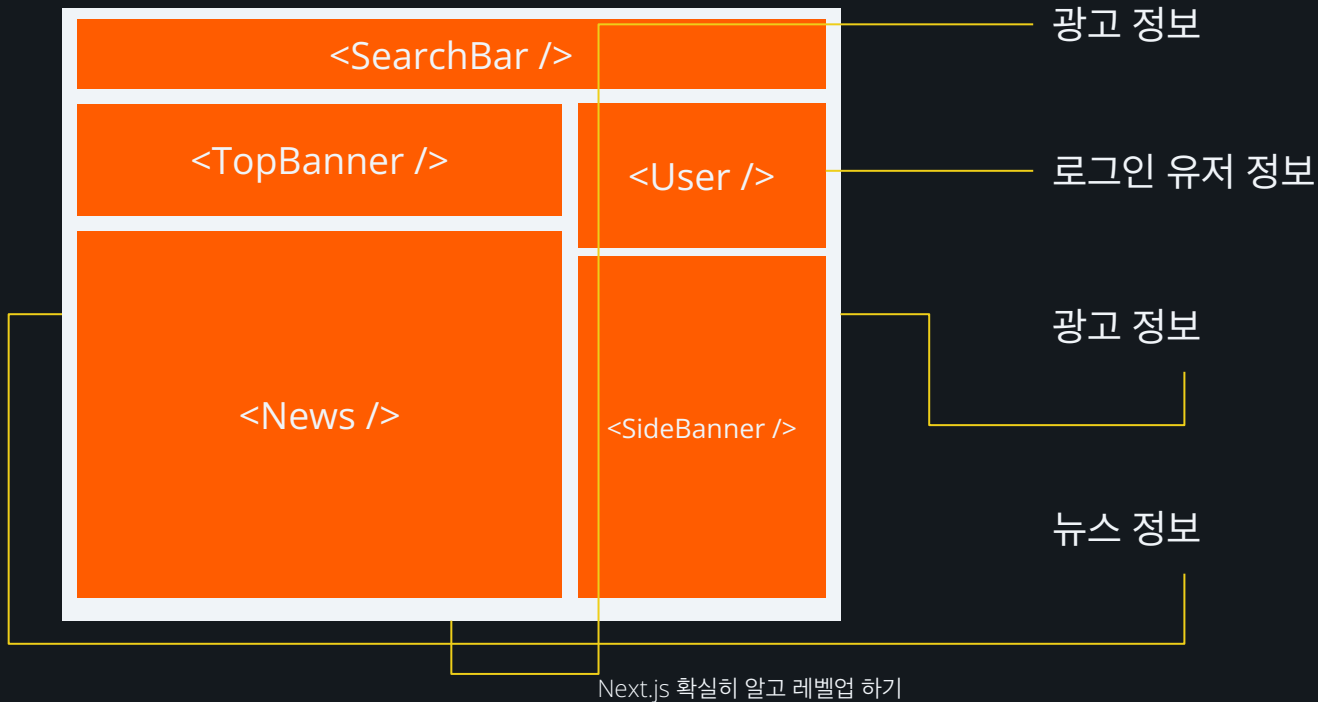
1. React 18 의 동시성

React 18 이전 버전의 문제점



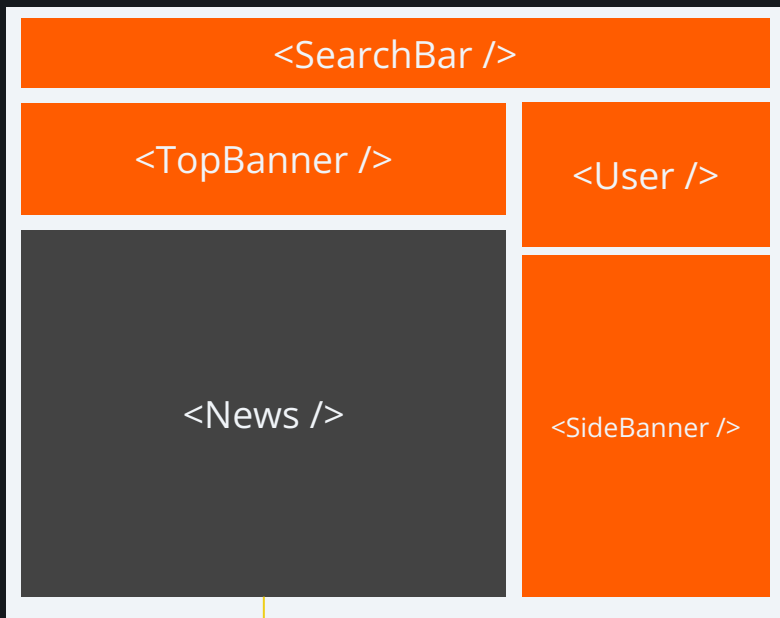
1. React 18 의 동시성

React 18 이전 버전의 문제점



1. React 18 의 동시성

React 18 이전 버전의 문제점

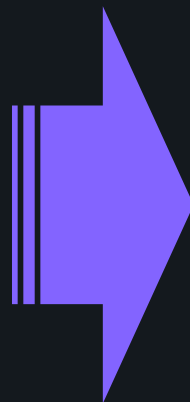
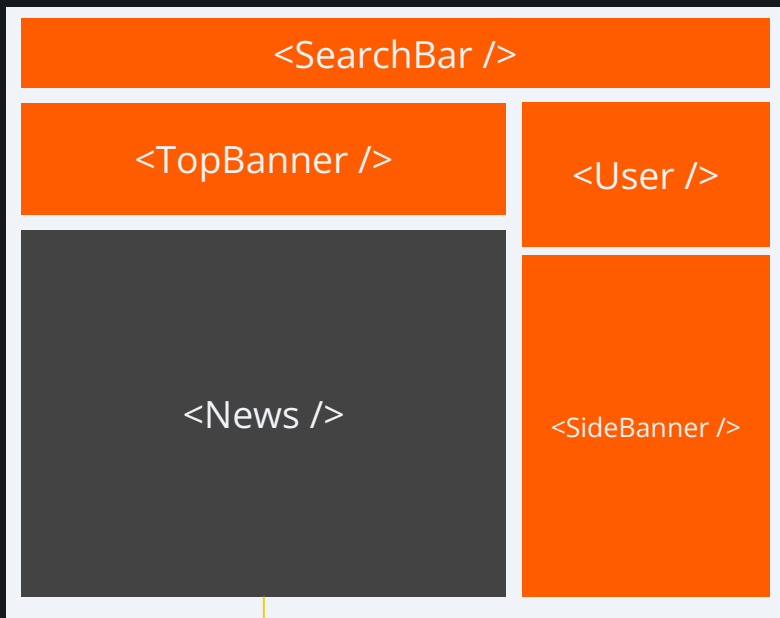


뉴스 정보

- 용량이 큰 JS
- 오래걸리는 API

1. React 18 의 동시성

React 18 이전 버전의 문제점



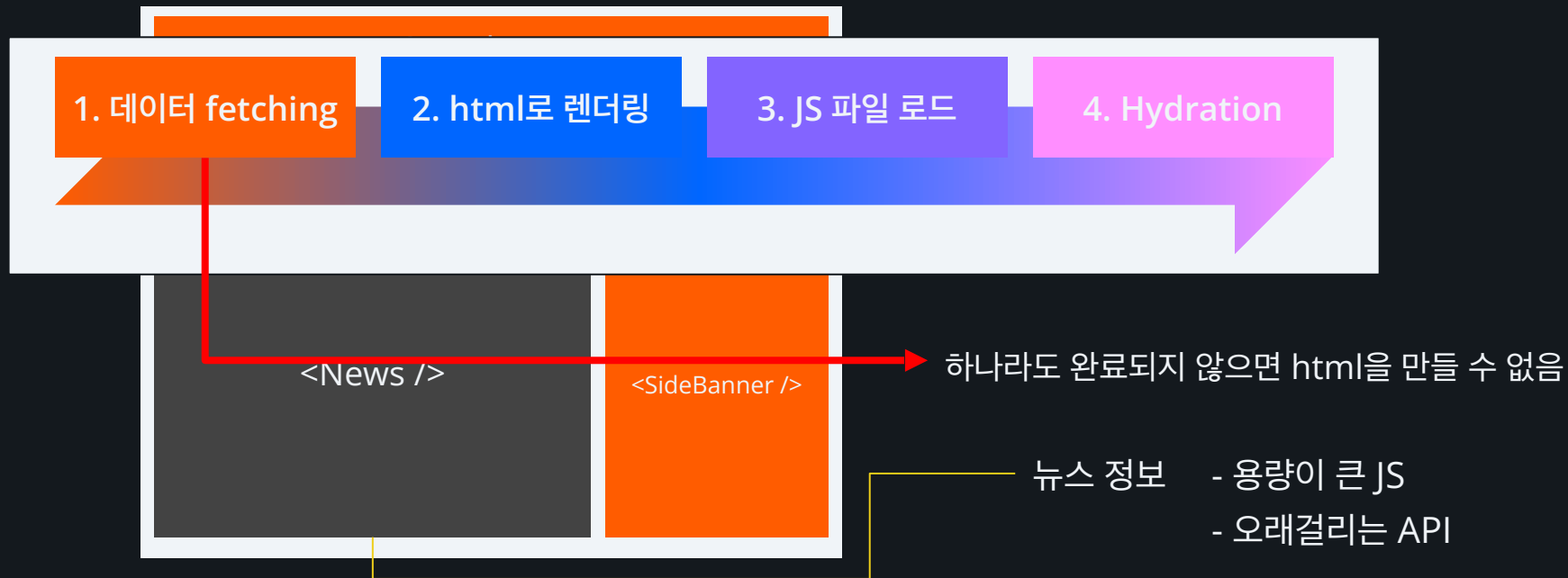
main.html

뉴스 정보

- 용량이 큰 JS
- 오래걸리는 API

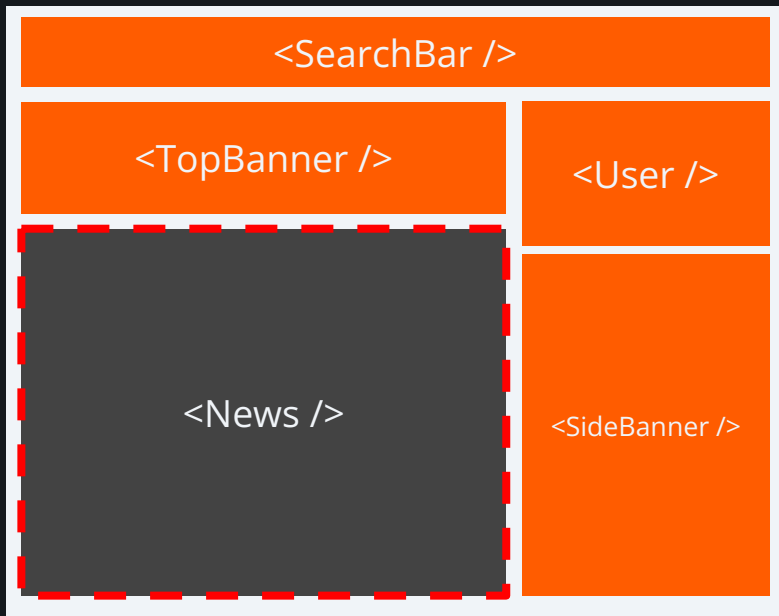
1. React 18 의 동시성

React 18 이전 버전의 문제점



1. React 18 의 동시성

React 18 이전 버전의 문제점



이것만 초기 렌더링에서 빼면 안되나요?

안 되는 건 아니지만... 🙄



1. React 18 의 동시성

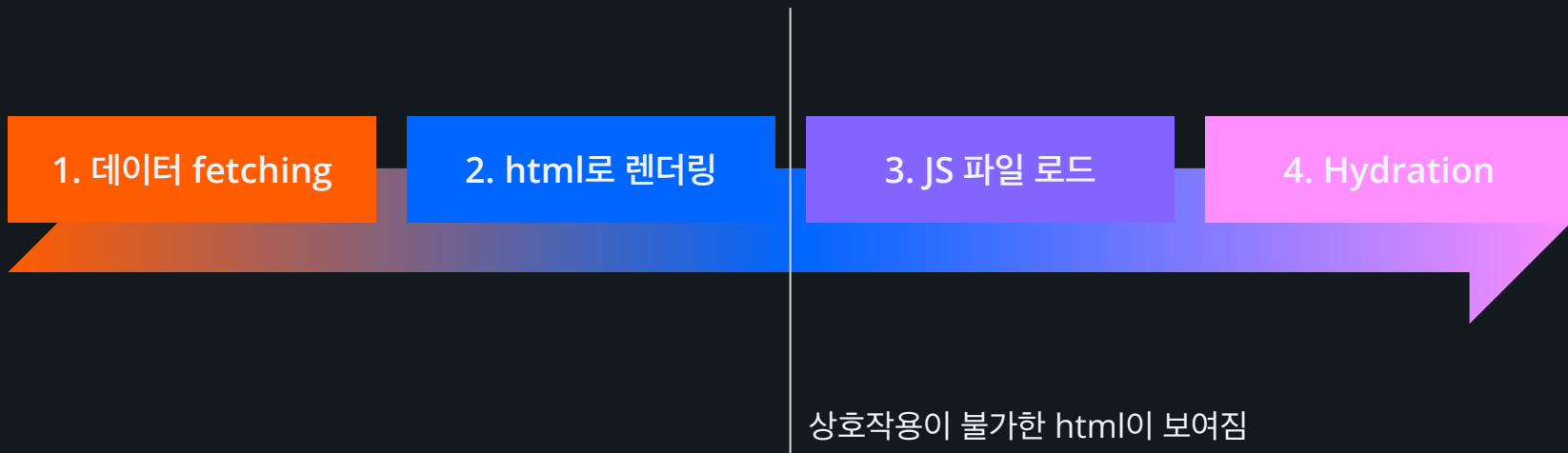
React 18 이전 버전의 문제점

모든 JS가 로드되기 전 까지는

Hydrate 할 수 없다.

1. React 18 의 동시성

React 18 이전 버전의 문제점





1. React 18 의 동시성

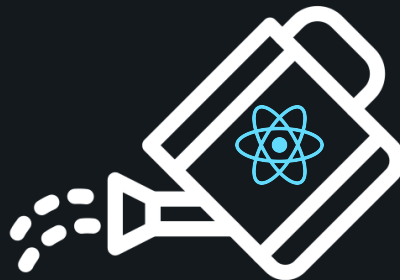
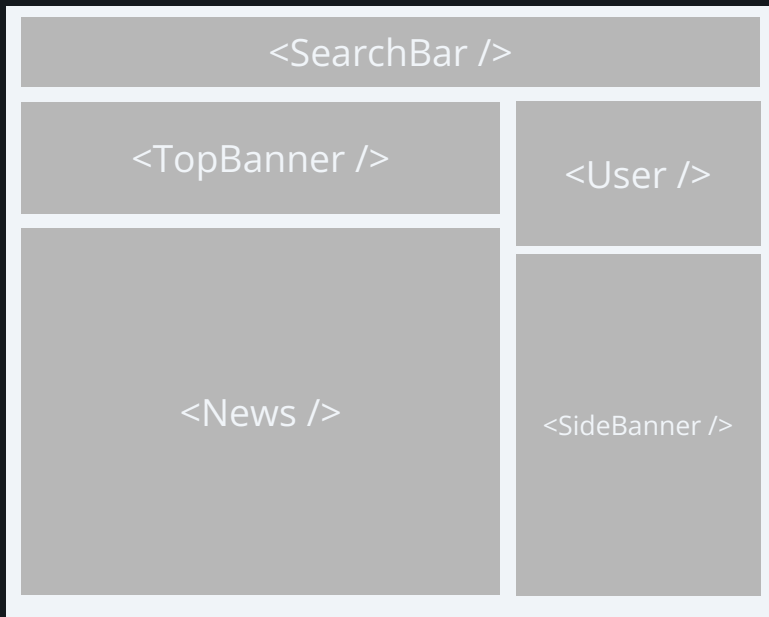
React 18 이전 버전의 문제점

`chrome://settings/content/javascript`

`www.wanted.co.kr`

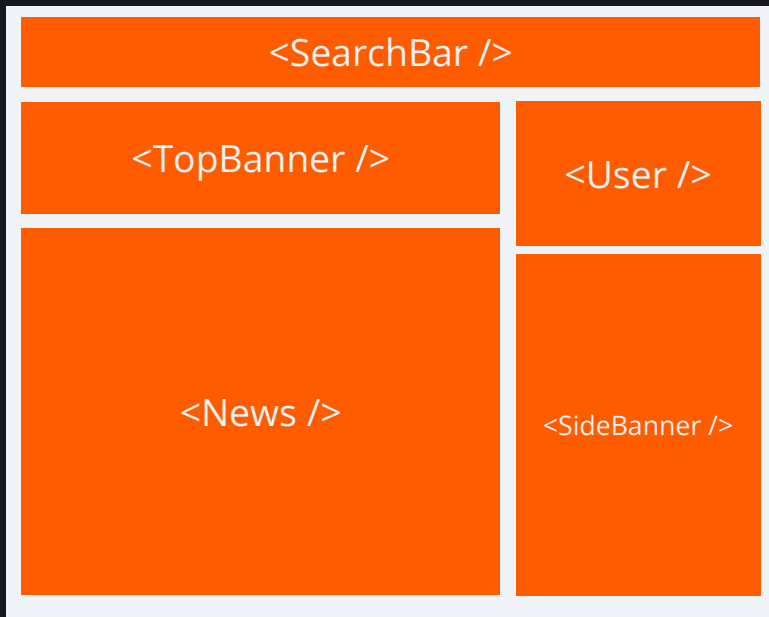
1. React 18 의 동시성

React 18 이전 버전의 문제점



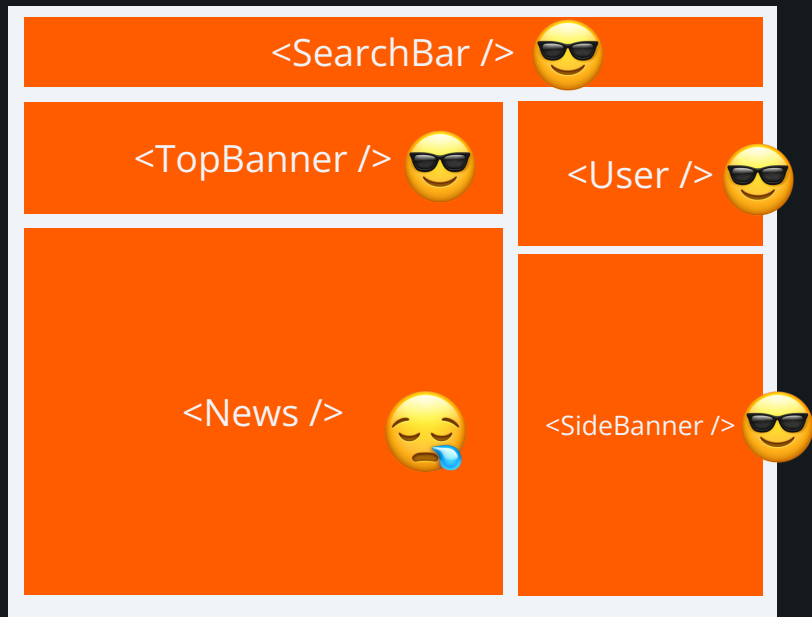
1. React 18 의 동시성

React 18 이전 버전의 문제점



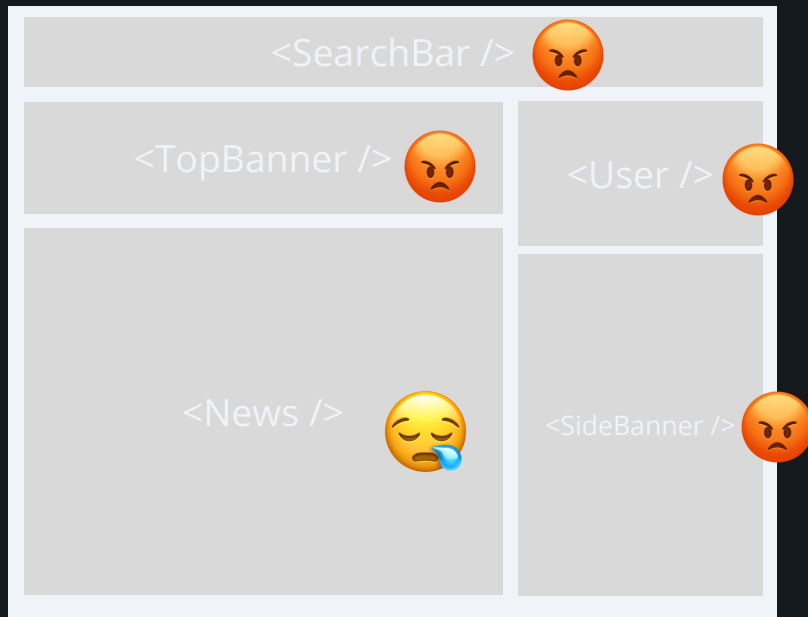
1. React 18 의 동시성

React 18 이전 버전의 문제점



1. React 18 의 동시성

React 18 이전 버전의 문제점





1. React 18 의 동시성

React 18 가 해결한 방법

Streaming
HTML



Suspense

1. React 18 의 동시성

React 18 가 해결한 방법

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

전체를 한번에 진행

1. React 18 의 동시성

React 18 가 해결한 방법

<SearchBar />

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

<News />

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

<TopBanner />

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

2. RSC 개념 (React Server Component)

2. React Server Component

Server Side Rendering

Ssr

SSR을 이용해보려고

SSR 사용을 위해서

SSR이 자격요건에서 많이 요구되고 있고 CSR을 이용한 프로젝트만 경험해봤기 때문에 학습하고 싶습니다.

SSR을 제대로 이해하고 싶어서

SSR 개념 파악 및 기술 트렌드 공부 목적

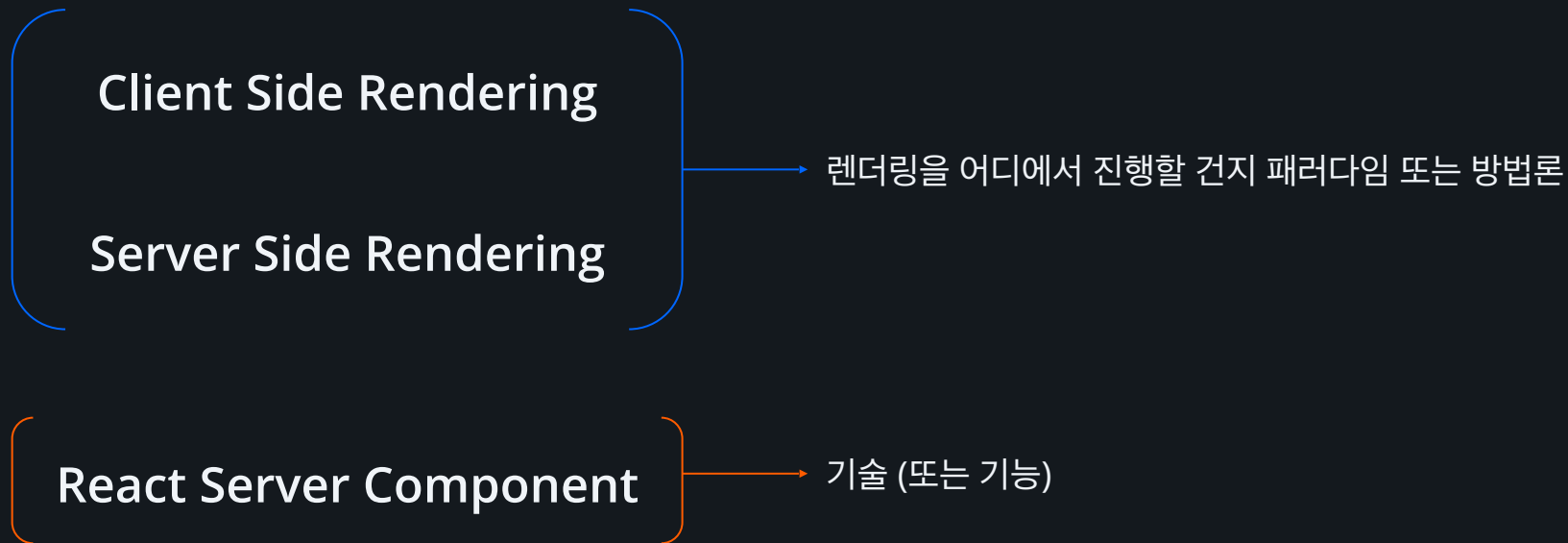
2. React Server Component


CSR, SSR, RSC ?



2. React Server Component

CSR, SSR, RSC ?






2. React Server Component

CSR, SSR, RSC ?

Server Side Rendering

React Server Component



2. React Server Component

React Server Component가 뭔가요?

서버에서 실행 되는 React 컴포넌트

2. React Server Component

React Server Component가 뭔가요?

서버에서 실행 되는 React 컴포넌트

기존에는 실행이 안된건가..?



2. React Server Component

SSR, SSR+RSC

SSR

- nextjs 서버에서 `getServerSideProps` 실행되며 데이터 fetching
 - 데이터 fetching 완료 후 컴포넌트 렌더링을 실행해 비대화형 html 생성
 - 클라이언트에서 js 로드 후 hydrate 과정을 거쳐 완벽한 html 완성
- 서버와 클라이언트에서 컴포넌트 실행 됨

2. React Server Component

SSR, SSR+RSC

SSR + RSC

- nextjs 서버에서 RSC 실행하며 데이터 fetching 및 렌더링 진행
- 렌더링 완료 후 생성 된 html를 클라이언트로 전달
- 클라이언트는 별도의 hydrate 과정 없이 전달 받은 html 사용

→ 서버에서만 컴포넌트 실행 됨

2. React Server Component

RSC의 장점



용량 감소

2. React Server Component

그럼 다 RSC로 개발하면 되는거 아닌가?

useState, useEffect, onClick 등 client 기능 사용 불가



2. React Server Component

RSC는 이벤트 핸들러 사용 불가

← → 1 of 1 error ● Next.js (15.1.6) (Turbopack) ✕

Unhandled Runtime Error



[Server] Error: Event handlers cannot be passed to Client Component props.

```
<ul className="mt-4" onClick={function onClick} children=...>
```

^^^^^^^^^^^^^^^^^^^^

If you need interactivity, consider converting part of this to a Client Component.

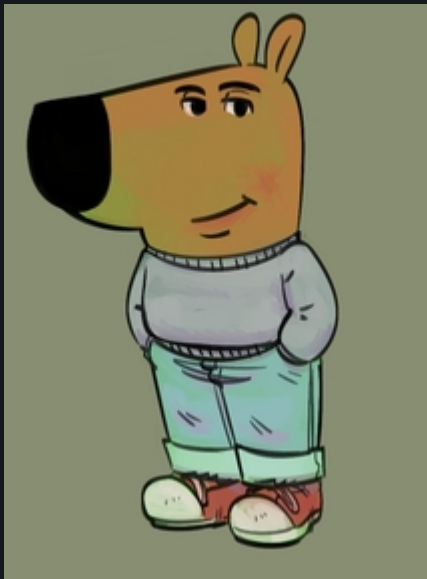
Show ignored frames

Nextjs: 님 이거 쓰려면 Client Component 쓰셈

2. React Server Component

RSC는 이벤트 핸들러 사용 불가

상호작용은 1도 안되지만 빠른 로딩을 성공해 만족한 chill guy
(곧 회사 짤림)



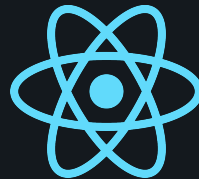
2. React Server Component

RSC는 이벤트 핸들러 사용이 왜 안될까?




js 없이 client 의 DOM 을 처리할 수 없다

onClick, onChange, ...etc



그런게 있었음?

Client React



2. React Server Component

RSC 만의 규칙

직렬화 (serialization) 가능 한 값만 전달 할 수 있다

2. React Server Component

직렬화(serialization) 란?

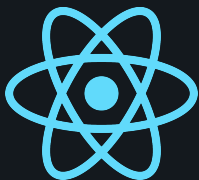
데이터를 다른 어딘가로 전송 하기 위해 특정 형식으로 변환하는 절차

JSON.parse( JSON 문자열을 객체(Object)로 변환
)

JSON.stringify( 객체(Object)를 JSON 문자열로 변환
)

2. React Server Component

직렬화(serialization) 란?



React 만의 직렬화 방식을 사용 (JSON 과 유사)

함수(function)은 직렬화 불가.

2. React Server Component

직렬화 예시

```
<div style={{ backgroundColor: 'blue' }}>
  Server Component
</div>;
```



```
{
  "$$typeof": "react.element",
  "type": "div",
  "props": {
    "style": {
      "backgroundColor": "blue"
    },
    "children": "Server Component"
  },
  "key": null,
  "ref": null,
  "_owner": null,
  "_store": {}
}
```


2. React Server Component

역직렬화 (deserialization)

```
{
  "$$typeof": "react.element",
  "type": "div",
  "props": {
    "style": {
      "backgroundColor": "blue"
    },
    "children": "Server Component"
  },
  "key": null,
  "ref": null,
  "_owner": null,
  "_store": {}
}
```



```
<div style={{ backgroundColor: 'blue' }}>
  Server Component
</div>;
```



2. React Server Component

함수는 왜 직렬화가 안되는가?

Execution Context 까지 직렬화 할 수 없기 때문

2. RSC 이점

3. React Server Component 이점

CSR, SSR, SSR+RSC

CSR

1. JS 파일 로드

2. 데이터 fetching

3. 컴포넌트 렌더링

4. 사용자 상호작용

상호작용 할 수 없는 html 을 보다 빨리 보여주어 UX 개선

SSR

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

3. React Server Component 이점

CSR, SSR, SSR+RSC

<SearchBar />

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

<News />

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

<TopBanner />

1. 데이터 fetching

2. html로 렌더링

3. JS 파일 로드

4. Hydration

3. React Server Component 이점

CSR, SSR, SSR+RSC

서버에서도 지원



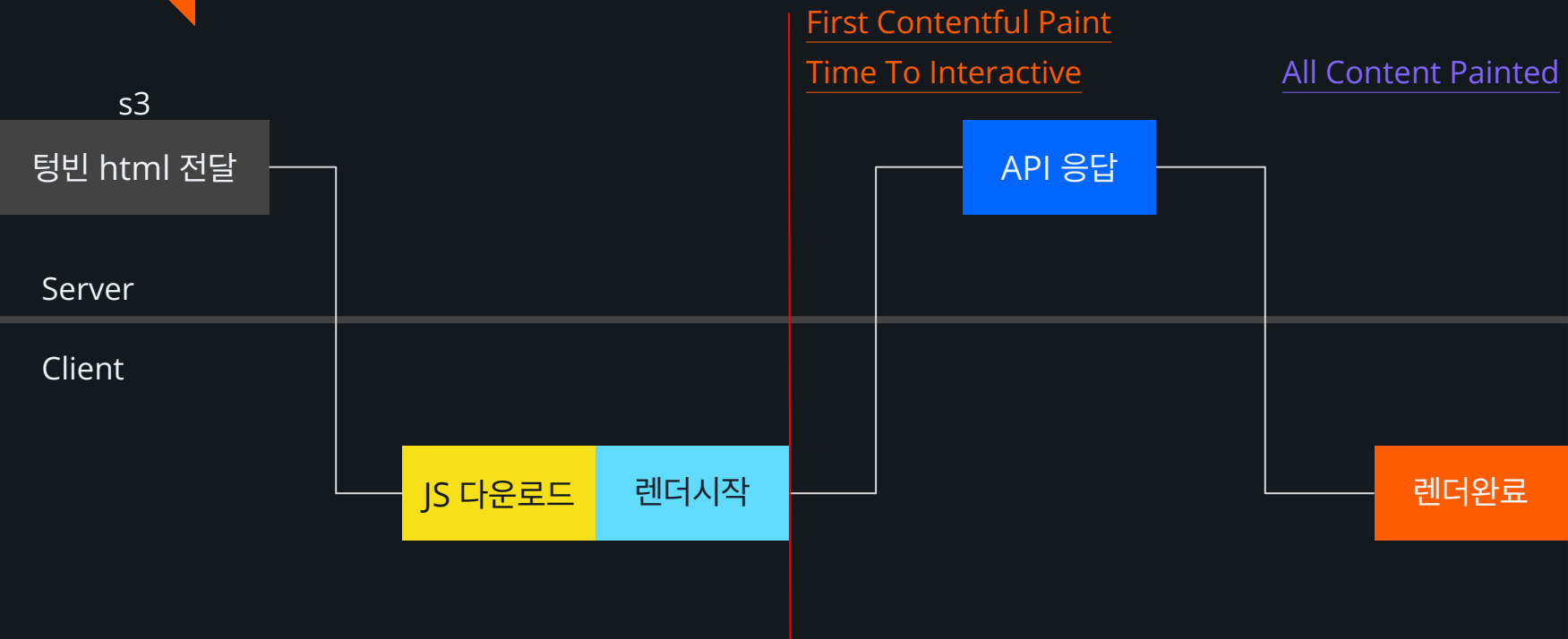
3. React Server Component 이점

CSR, SSR, SSR+RSC



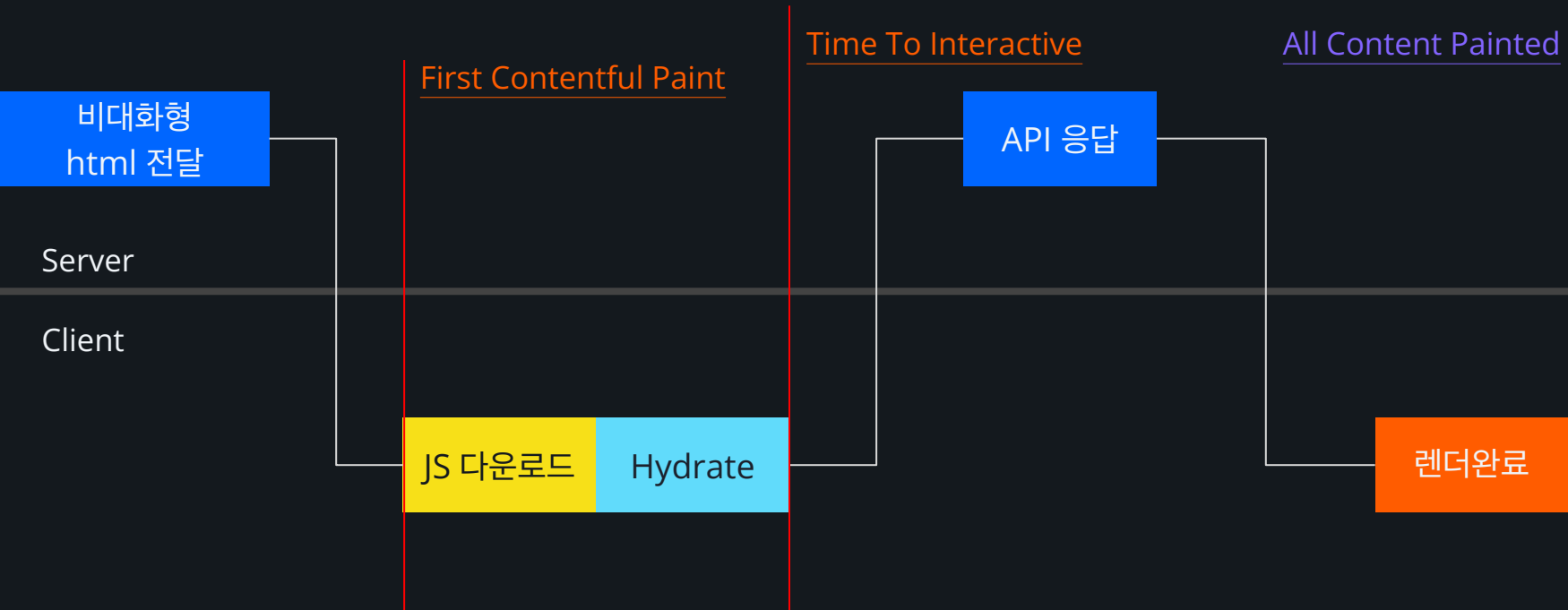
3. React Server Component 이점

CSR, SSR, SSR+RSC



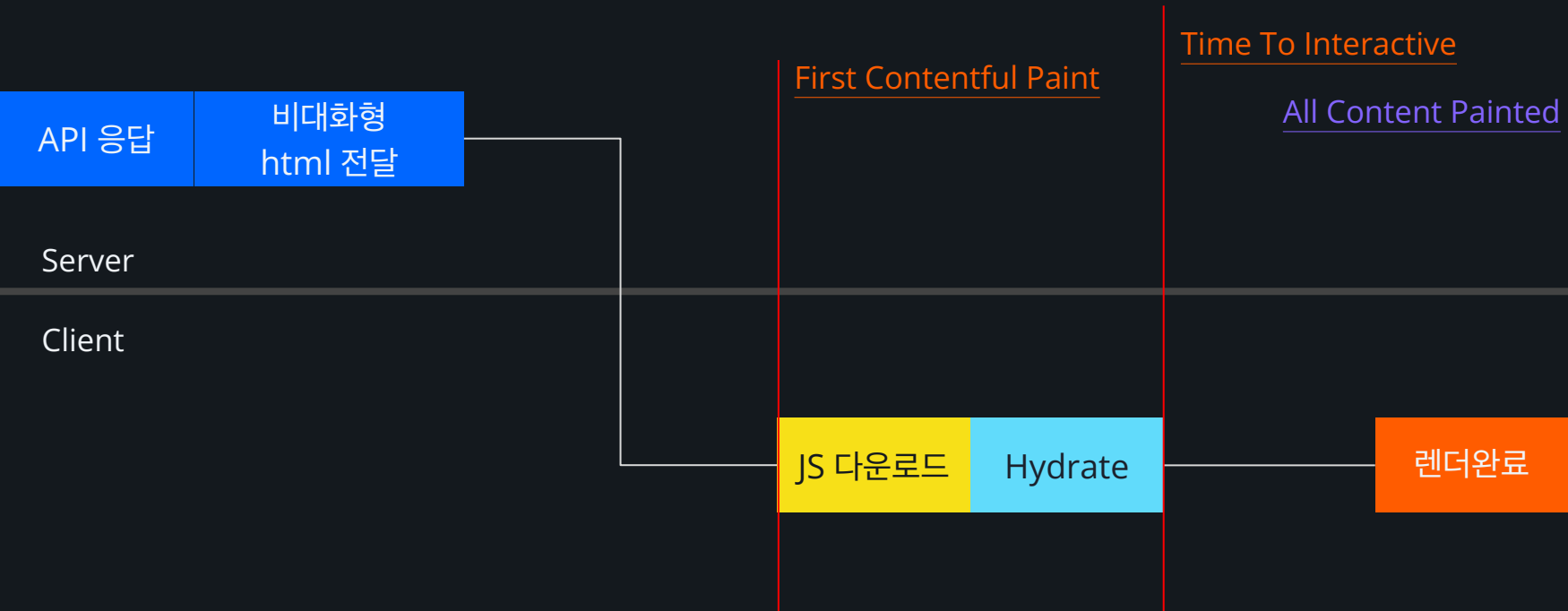
3. React Server Component 이점

CSR, SSR, SSR+RSC



3. React Server Component 이점

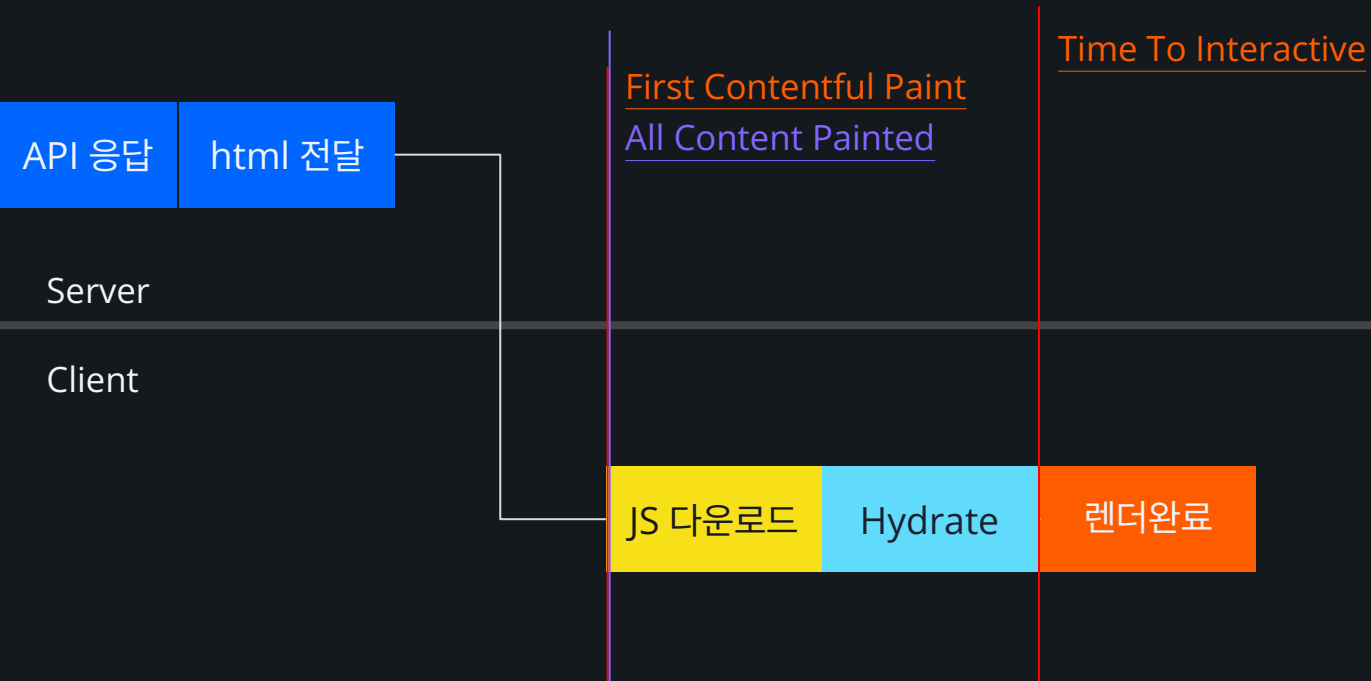
CSR, SSR, SSR+RSC



3. React Server Component 이점

SSR+RSC

CSR, SSR, SSR+RSC

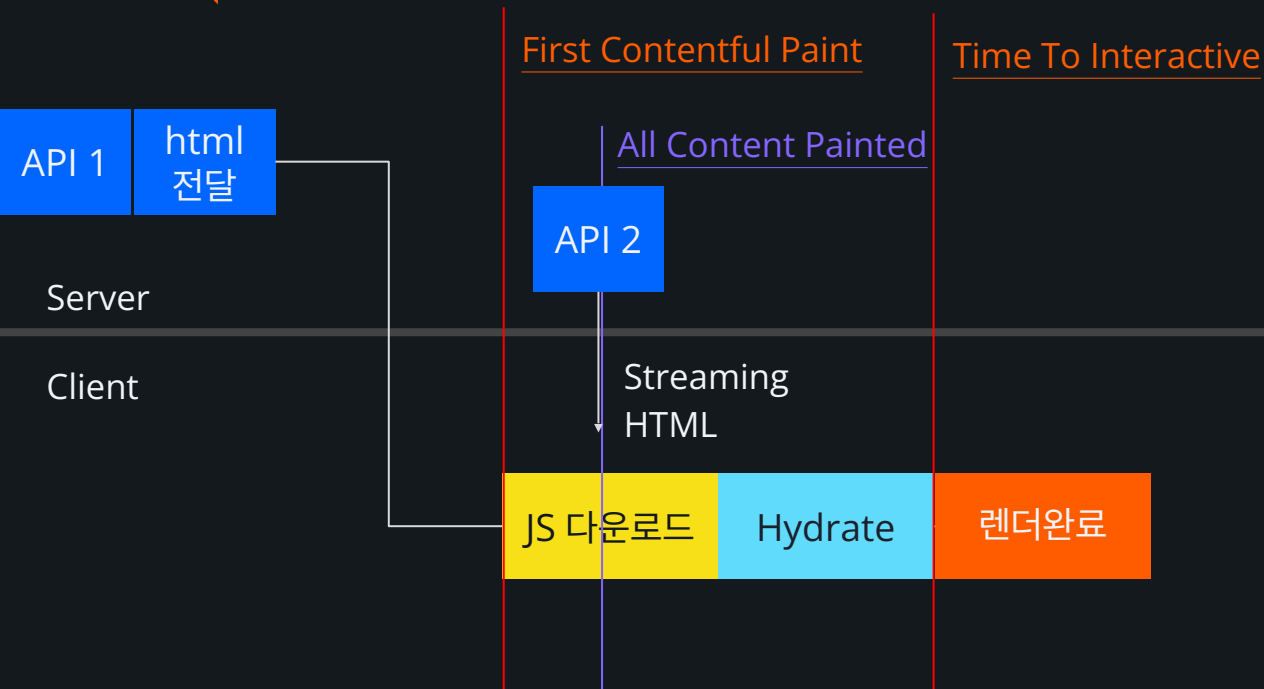


3. React Server Component 이점

CSR, SSR, SSR+RSC

SSR+RSC

w. Suspense



고생하셨습니다.