resumeAndPrerenderToNodeStream

`resumeAndPrerenderToNodeStream` continues a prerendered React tree to a static HTML string using a a [Node.js Stream.](#).

```
const {prelude, postponed} = await resumeAndPrerenderToNodeStream(reactNo
```

- [Reference](#reference)
  - [`resumeAndPrerenderToNodeStream(reactNode, postponedState, options?)`](#)
- [Usage](#usage)
  - [Further reading](#further-reading)

> ### ⊟ 중요합니다!
>
> This API is specific to Node.js. Environments with [Web Streams,](#) like Deno and modern edge runtimes, should use `prerender` instead.

Reference

`resumeAndPrerenderToNodeStream(reactNode, postponedState, options?)`

Call `resumeAndPrerenderToNodeStream` to continue a prerendered React tree to a static HTML string.

```
import { resumeAndPrerenderToNodeStream } from 'react-dom/static';
import { getPostponedState } from 'storage';

async function handler(request, writable) {
  const postponedState = getPostponedState(request);
  const { prelude } = await resumeAndPrerenderToNodeStream(<App />, JSON.parse(postpon
  prelude.pipe(writable);
}
```

On the client, call `hydrateRoot` to make the server-generated HTML interactive.

See more examples below.

Parameters

- `reactNode` : The React node you called `prerender` (or a previous `resumeAndPrerenderToNodeStream`) with. For example, a JSX element like `<App />`. It is expected to represent the entire document, so the `App` component should render the `<html>` tag.

- `postponedState` : The opaque `postpone` object returned from a prerender API, loaded from wherever you stored it (e.g. redis, a file, or S3).

- **optional** `options` : An object with streaming options.

  - **optional** `signal` : An abort signal that lets you abort server rendering and render the rest on the client.

  - **optional** `onError` : A callback that fires whenever there is a server error, whether recoverable or not. By default, this only calls `console.error`. If you override it to log crash reports, make sure that you still call `console.error`.

Returns

`resumeAndPrerenderToNodeStream` returns a Promise:

- If rendering the is successful, the Promise will resolve to an object containing:

- `prelude`: a [Web Stream](#) of HTML. You can use this stream to send a response in chunks, or you can read the entire stream into a string.
- `postponed`: an JSON-serializeable, opaque object that can be passed to `resumeToNodeStream` or `resumeAndPrerenderToNodeStream` if `resumeAndPrerenderToNodeStream` is aborted.
- If rendering fails, the Promise will be rejected. [Use this to output a fallback shell.](#)

Caveats

`nonce` is not an available option when prerendering. Nonces must be unique per request and if you use nonces to secure your application with [CSP](#) it would be inappropriate and insecure to include the nonce value in the prerender itself.

📄 **중요합니다!**

When should I use `resumeAndPrerenderToNodeStream`?

The static `resumeAndPrerenderToNodeStream` API is used for static server-side generation (SSG). Unlike `renderToString`, `resumeAndPrerenderToNodeStream` waits for all data to load before resolving. This makes it suitable for generating static HTML for a full page, including data that needs to be fetched using Suspense. To stream content as it loads, use a streaming server-side render (SSR) API like [renderToReadableStream](#).

`resumeAndPrerenderToNodeStream` can be aborted and later either continued with another `resumeAndPrerenderToNodeStream` or resumed with `resume` to support partial pre-rendering.

Usage

Further reading

`resumeAndPrerenderToNodeStream` behaves similarly to `prerender` but can be used to continue a previously started prerendering process that was aborted.
For more information about resuming a prerendered tree, see the resume documentation.

Meta Open Source

Copyright © Meta Platforms, Inc

uwu?

React 학습하기

빠르게 시작하기

설치하기

UI 표현하기

상호작용성 더하기

State 관리하기

탈출구

API 참고서

React APIs

React DOM APIs

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관