resumeToPipeableStream

`resumeToPipeableStream` streams a pre-rendered React tree to a pipeable Node.js Stream.

```
const {pipe, abort} = await resumeToPipeableStream(reactNode, postponedSt
```

중요합니다!

This API is specific to Node.js. Environments with Web Streams, like Deno and modern edge runtimes, should use `resume` instead.

Reference

`resumeToPipeableStream(node, postponed, options?)`

Call `resume` to resume rendering a pre-rendered React tree as HTML into a Node.js Stream.

```
import { resume } from 'react-dom/server';
import {getPostponedState} from './storage';

async function handler(request, response) {
  const postponed = await getPostponedState(request);
  const {pipe} = resumeToPipeableStream(<App />, postponed, {
    onShellReady: () => {
      pipe(response);
    }
  });
}
```

See more examples below.

Parameters

- `reactNode`: The React node you called `prerender` 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** `nonce`: A `nonce` string to allow scripts for `script-src` Content-Security-Policy.
  - **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`.
  - **optional** `onShellReady`: A callback that fires right after the shell has finished. You can call `pipe` here to start streaming. React will stream the additional content after the shell along with the inline `<script>` tags that replace the HTML loading fallbacks with the content.
  - **optional** `onShellError`: A callback that fires if there was an error rendering the shell. It receives the error as an argument. No bytes were emitted from the stream yet, and neither `onShellReady` nor `onAllReady` will get called, so you can output a fallback HTML shell or use the prelude.

Returns

`resume` returns an object with two methods:

- `pipe` outputs the HTML into the provided Writable Node.js Stream. Call `pipe` in `onShellReady` if you want to enable streaming, or in `onAllReady` for crawlers and static generation.
- `abort` lets you abort server rendering and render the rest on the client.

Caveats

- `resumeToPipeableStream` does not accept options for `bootstrapScripts`, `bootstrapScriptContent`, or `bootstrapModules`. Instead, you need to pass these options to the `prerender` call that generates the `postponedState`. You can also inject bootstrap content into the writable stream manually.
- `resumeToPipeableStream` does not accept `identifierPrefix` since the prefix needs to be the same in both `prerender` and `resumeToPipeableStream`.
- Since `nonce` cannot be provided to prerender, you should only provide `nonce` to `resumeToPipeableStream` if you're not providing scripts to prerender.
- `resumeToPipeableStream` re-renders from the root until it finds a component that was not fully pre-rendered. Only fully prerendered Components (the Component and its children finished prerendering) are skipped entirely.

Usage

Further reading

Resuming behaves like `renderToReadableStream`. For more examples, check out the usage section of `renderToReadableStream`.
The usage section of `prerender` includes examples of how to use `prerenderToNodeStream` specifically.

이전
resume

다음
정적 API

Meta Open Source

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