

RenderToReadableStreamOptions

August 18, 2026

RenderToReadableStreamOptions

Kind: Interface

Source: `src/jsx/dom/server.ts`

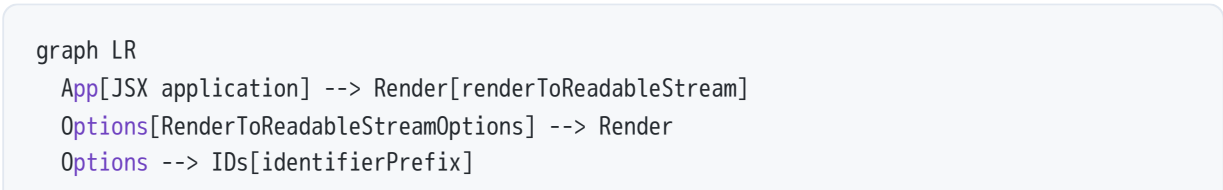
Part of: `Jsx`

`RenderToReadableStreamOptions` configures how server-rendered JSX is emitted as a readable stream. It controls identifier and namespace settings, bootstrap scripts, chunking behavior, cancellation through an `AbortSignal`, and error handling.

Properties

Property	Type
<code>identifierPrefix</code>	<code>string</code>
<code>namespaceURI</code>	<code>string</code>
<code>nonce</code>	<code>string</code>
<code>bootstrapScriptContent</code>	<code>string</code>
<code>bootstrapScripts</code>	<code>string[]</code>
<code>bootstrapModules</code>	<code>string[]</code>
<code>progressiveChunkSize</code>	<code>number</code>
<code>signal</code>	<code>AbortSignal</code>
<code>onError</code>	<code>`(error: unknown) => string`</code>

Diagram



```
Options --> Namespace[namespaceURI]
Options --> Scripts[Bootstrap scripts and modules]
Options --> Stream[progressiveChunkSize]
Options --> Abort[signal]
Options --> Errors[onError]
Render --> Output[ReadableStream]
```

Usage

```
import { renderToReadableStream } from "react-dom/server";
import type { RenderToReadableStreamOptions } from "react-dom/server";

const controller = new AbortController();

const options: RenderToReadableStreamOptions = {
  identifierPrefix: "app-",
  namespaceURI: "http://www.w3.org/2000/svg",
  nonce: "request-nonce",
  bootstrapScriptContent: "window.__SSR_READY__ = true;",
  bootstrapScripts: ["/assets/client.js"],
  bootstrapModules: ["/assets/client.mjs"],
  signal: controller.signal,
  onError(error) {
    console.error("Server render failed:", error);
    return "render-error";
  },
};

const stream = await renderToReadableStream(<App />, options);

return new Response(stream, {
  headers: {
    "Content-Type": "text/html",
  },
});
```

AI Coding Instructions

- Pass a stable `identifierPrefix` when multiple rendered roots can appear in the same document.
- Set `nonce` when the page uses a Content Security Policy that restricts inline scripts.
- Use either `bootstrapScripts` or `bootstrapModules` according to the client asset format being loaded.

- Pass a request-scoped `AbortSignal` so rendering stops when the client disconnects or the request is cancelled.
- Handle errors in `onError` ; return a string when the renderer needs an error identifier, or return `void` after logging the error.

How it works

`RenderToReadableStreamOptions` is the exported TypeScript interface for the optional second argument to `renderToReadableStream` ; every member is optional.

[src/jsx/dom/server.ts:32-42](#) [src/jsx/dom/server.ts:50-53](#)

Its declared members are:

- `identifierPrefix?: string`
- `namespaceURI?: string`
- `nonce?: string`
- `bootstrapScriptContent?: string`
- `bootstrapScripts?: string[]`
- `bootstrapModules?: string[]`
- `progressiveChunkSize?: number`
- `signal?: AbortSignal`
- `onError?: (error: unknown) => string | void`

[src/jsx/dom/server.ts:32-42](#)

At runtime, `renderToReadableStream` checks the option object's keys. If it contains any key other than `onError` , it calls `console.warn('options are not supported yet, except onError')` . [src/jsx/dom/server.ts:54-56](#) The function forwards only `options.onError` to the underlying streaming renderer. [src/jsx/dom/server.ts:62](#)

Consequently, the source does not show runtime handling for `identifierPrefix` , `namespaceURI` , `nonce` , `bootstrapScriptContent` , `bootstrapScripts` , `bootstrapModules` , `progressiveChunkSize` , or `signal` ; their presence triggers the warning above.

[src/jsx/dom/server.ts:32-42](#) [src/jsx/dom/server.ts:54-56](#) [src/jsx/dom/server.ts:62](#)

`onError` is used by the underlying renderer for errors during stream startup and for rejected deferred callbacks. If absent, that renderer defaults it to `console.trace` . [src/jsx/streaming.ts:146-149](#) Rejected deferred callbacks are logged with `console.log` , passed to `onError` , and replaced with an empty string for subsequent processing. [src/jsx/streaming.ts:171-195](#) Errors caught while starting the stream are passed to

`onError` ; after the `try / catch` , the stream is closed unless it was cancelled.
[src/jsx/streaming.ts:152-167](#) [src/jsx/streaming.ts:203-217](#)