captureRenderContext

August 18, 2026

captureRenderContext

Kind: Function

Source: `src/jsx/context.ts`

Part of: Jsx

Capture the current render store and return a resumer that re-establishes it around a deferred continuation (e.g. a re-render after a suspended promise settles). Shared by every suspension point so none reimplements it.

`captureRenderContext` saves the render store that is active when it is called and returns a resumer function. Call the resumer around deferred work, such as a continuation after a suspended promise settles, so that work runs with the original render context.

Signature

```
function captureRenderContext(): (<T>(callback: () => T) => T)
```

Returns: `(<T>(callback: () => T) => T)`

Diagram

```
graph LR
  A[Active render store] --> B[captureRenderContext]
  B --> C[Resumer function]
  D[Deferred continuation] --> C
  C --> E[Restore captured render store]
  E --> F[Run continuation]
  F --> G[Restore previous render store]
```

Usage

```ts
import { captureRenderContext } from './context'

function continueAfterPromise(promise: Promise<void>, rerender: () => void) {
  const resume = captureRenderContext()

  promise.then(() => {
    resume(() => {
      rerender()
    })
  })
}
```

AI Coding Instructions

- Capture the render context before scheduling asynchronous or deferred work.
- Invoke deferred continuations through the returned resumer rather than calling them directly.
- Preserve the existing render store after the continuation completes; do not assign global render state without restoring it.
- Share this helper across suspension paths so promise handlers and retry logic follow the same context-restoration behavior.

Used by

3 references from 3 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (3)

- `Props` — `src/jsx/base.ts` :27
- `childrenToString` — `src/jsx/components.ts` :14
- `StreamingContext` — `src/jsx/streaming.ts` :30