Root

August 19, 2026

Root

Kind: Interface

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

Part of: Jsx

`Root` represents a mounted JSX DOM tree created by the client renderer. It exposes methods to render the tree and to remove the mounted tree from its container.

Diagram

```
graph LR
  App[JSX application] --> Root[Root]
  Root --> Render[render()]
  Root --> Unmount[unmount()]
  Render --> DOM[DOM container]
  Unmount --> DOM
```

Usage

```
import { createRoot } from "./jsx/dom/client";

const container = document.getElementById("app");

if (!container) {
  throw new Error("Missing app container");
}

const root = createRoot(container);

root.render();

window.addEventListener("beforeunload", () => {
  root.unmount();
});
```

AI Coding Instructions

- Call `render()` after creating a `Root` to mount or update its JSX DOM tree.
- Call `unmount()` when the owning application or view is removed.
- Keep the `Root` instance associated with the DOM container it was created for.
- Do not call `render()` after `unmount()` unless the client renderer supports remounting.

How it works

`Root` is the interface for the object returned by `createRoot`. It has two void-returning lifecycle methods: `render(children: Child)` and `unmount()` (src/jsx/dom/client.ts:11-14). `Child` includes strings, numbers, JSX nodes, `null`, `undefined`, booleans, promises of strings, and nested child arrays (src/jsx/base.ts:162-170).

- **Creation:** `createRoot` accepts an `HTMLElement` or `DocumentFragment` render target and returns a `Root` (src/jsx/dom/client.ts:23-26). Passing a non-empty options object writes `createRoot options are not supported yet` to `console.warn` (src/jsx/dom/client.ts:33-35).
- `render(children)`**:** On its first call, the root creates a function node whose state is initialized from the passed value; it retains that state setter for later calls (src/jsx/dom/client.ts:47-58). The DOM renderer builds this node, applies it into a `DocumentFragment`, then replaces every child of the target container with that fragment (src/jsx/dom/render.ts:783-794). On later calls, `render` passes the new value to the retained state setter (src/jsx/dom/client.ts:43-45); the setter changes state only when the new value is not `Object.is`-equal to the current value and then schedules an update (src/jsx/hooks/index.ts:199-240).
- `unmount()`**:** If the root has rendered, `unmount` calls its stored state setter with `null`, then marks the root as unmounted (src/jsx/dom/client.ts:61-64). A later `render` call throws `Error('Cannot update an unmounted root')` (src/jsx/dom/client.ts:38-42). Calling `unmount` before the first render does not invoke a setter, but still marks the root unmounted (src/jsx/dom/client.ts:27-31, src/jsx/dom/client.ts:61-64).

Relationships

- IMPORTS → `useState`
- IMPORTS → `buildNode`
- IMPORTS → `renderNode`