**Commit Phase**

**Commit Phase Overview:**

* React writes the computed updates from the render phase to the DOM.
* The commit phase is synchronous, ensuring the DOM is updated consistently.
* This prevents partial updates, maintaining UI consistency.
* The work-in-progress Fiber tree becomes the current tree for the next render cycle.

**React and React DOM:**

* React itself does not touch the DOM; it handles the render phase.
* React DOM, a separate library, commits the render phase results to the DOM.
* This design allows React to be used with various platforms, not just the web DOM (e.g., React Native, Remotion, etc.).

**High-Level Overview of Rendering Process**

**Triggering Renders:**

* Initial render or state updates trigger the render phase.

**Render Phase:**

* React calls component functions to create updated React elements, forming a new virtual DOM.
* All child components re-render when a parent component re-renders.
* The new virtual DOM is reconciled with the current Fiber tree to find necessary DOM updates.
* The Fiber tree, a mutable data structure, tracks component state, props, and effects.
* Reconciliation uses a diffing algorithm to compute the smallest number of DOM updates.

**Output of Render Phase:**

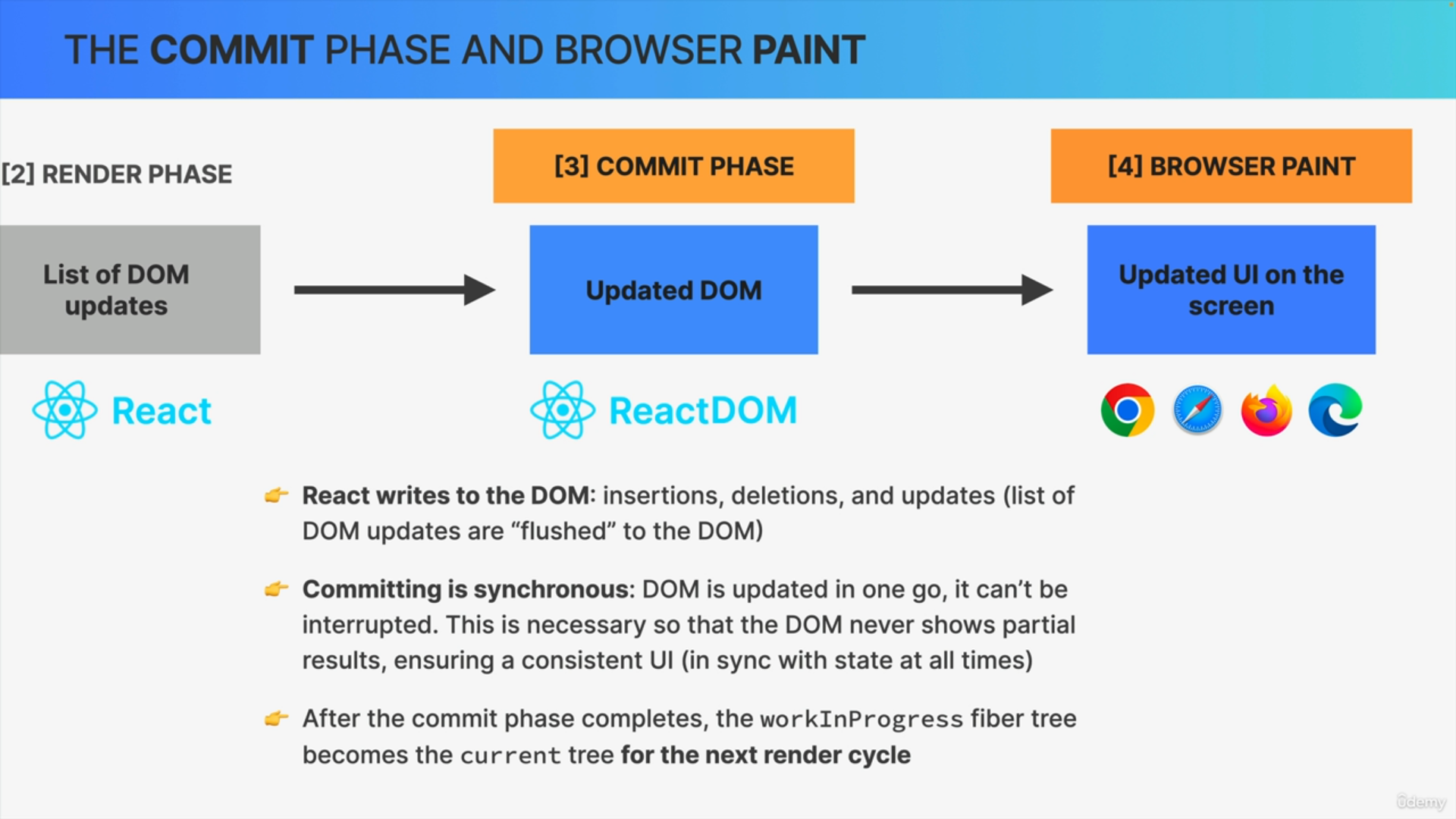
* An updated Fiber tree and a list of necessary DOM updates are produced.
* The render phase is asynchronous, allowing work to be prioritized, split into chunks, and resumed later.

**Commit Phase:**

* React DOM (or another renderer) applies the computed DOM updates synchronously.
* This ensures a consistent and accurate update of the UI.

**Browser Paint:**

* The browser detects DOM changes and repaints the screen, updating the user interface.



**Key Points**

**React Library:**

* Only handles the render phase, not the commit phase.
* Designed to be platform-agnostic, supporting various hosts beyond the web DOM.

**Practical Implications:**

* Understanding the rendering process aids in optimizing performance and dealing with practical implications in React applications.

**Conclusion:**

* Knowing these details helps deepen the understanding of React's inner workings.
* Practical knowledge of this process can lead to better performance and more efficient React applications.

