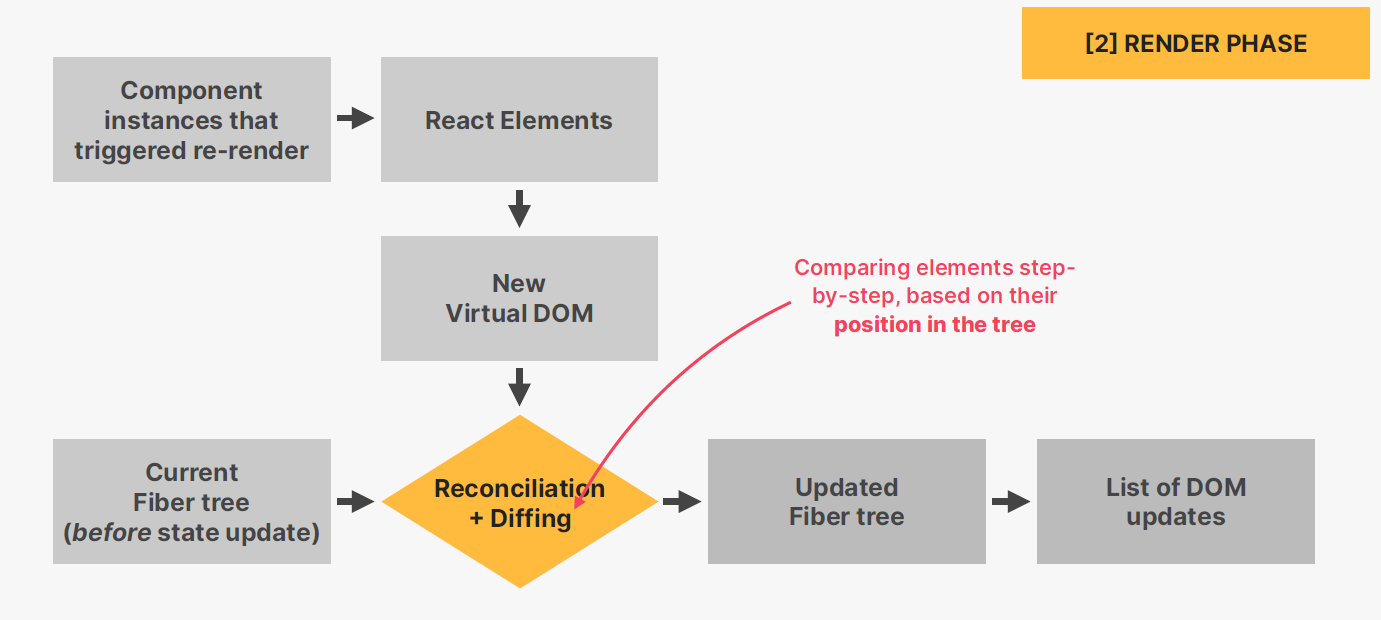
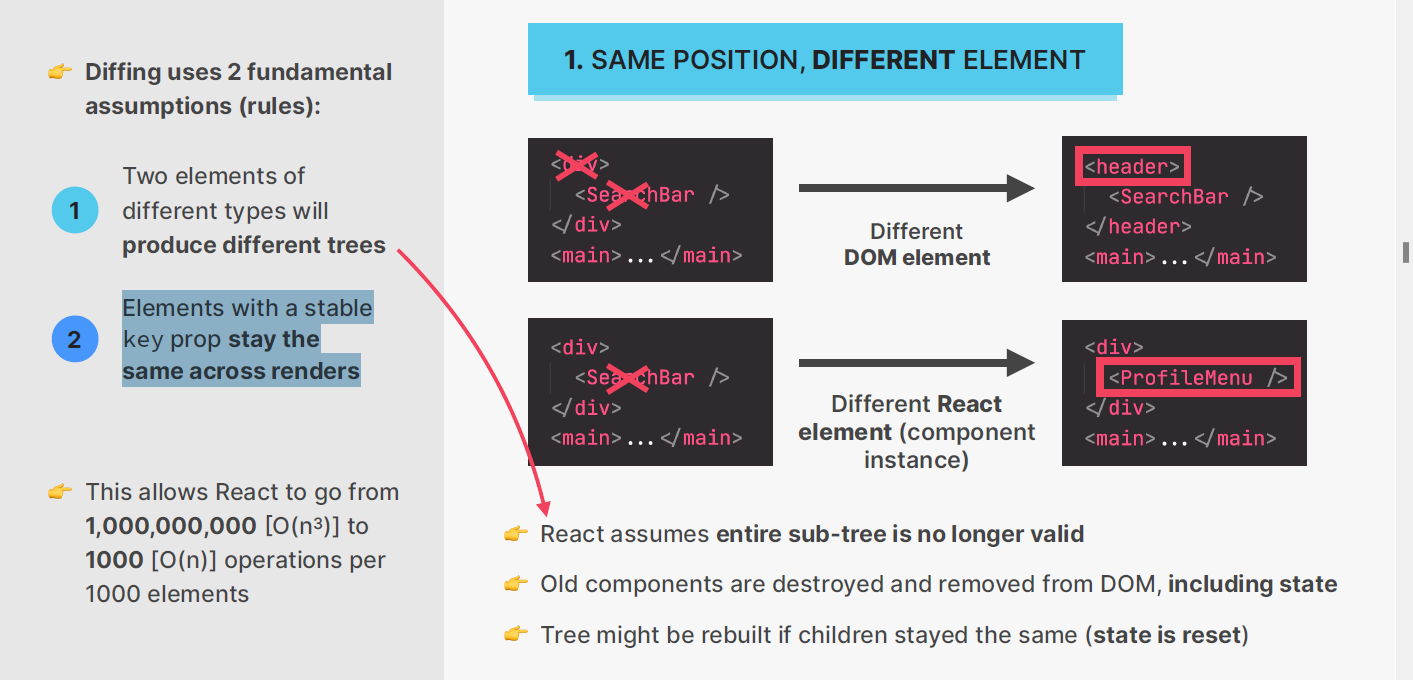
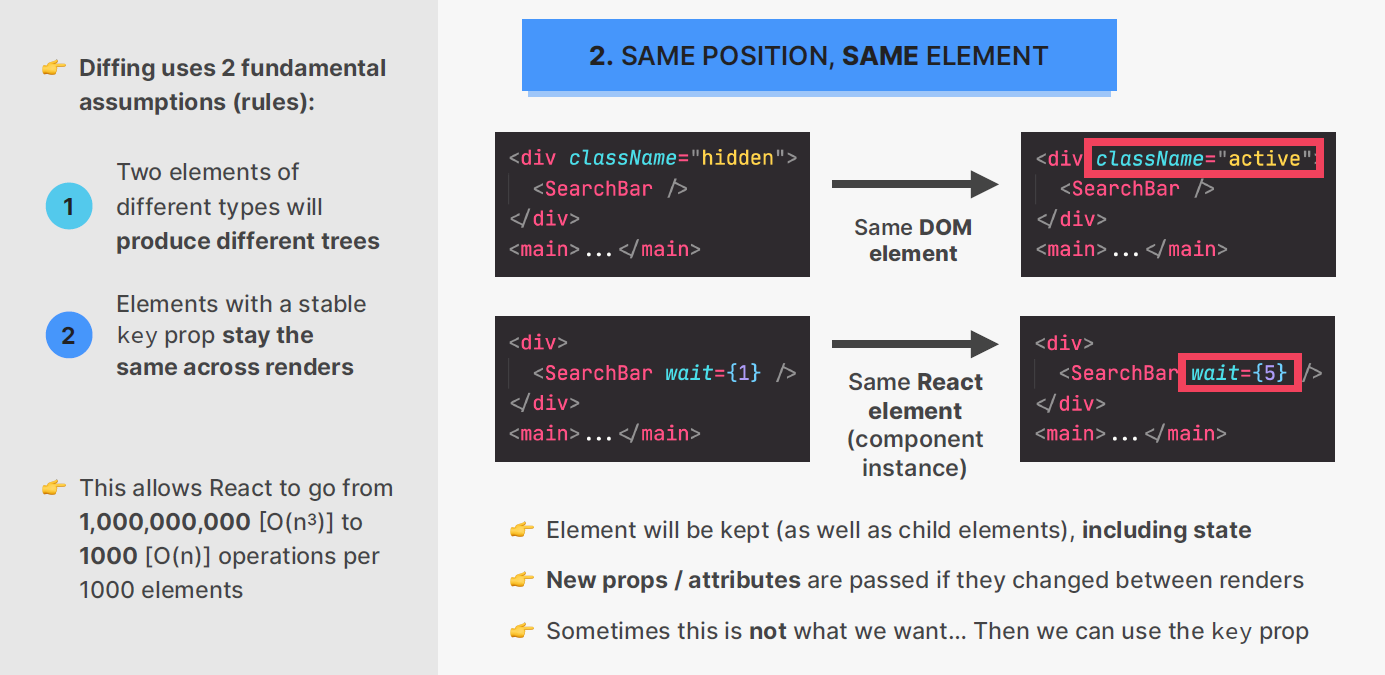
**DIFFING**







**TWO ELEMENTS OF DIFFERENT TYPES WILL PRODUCE DIFFERENT TREES**

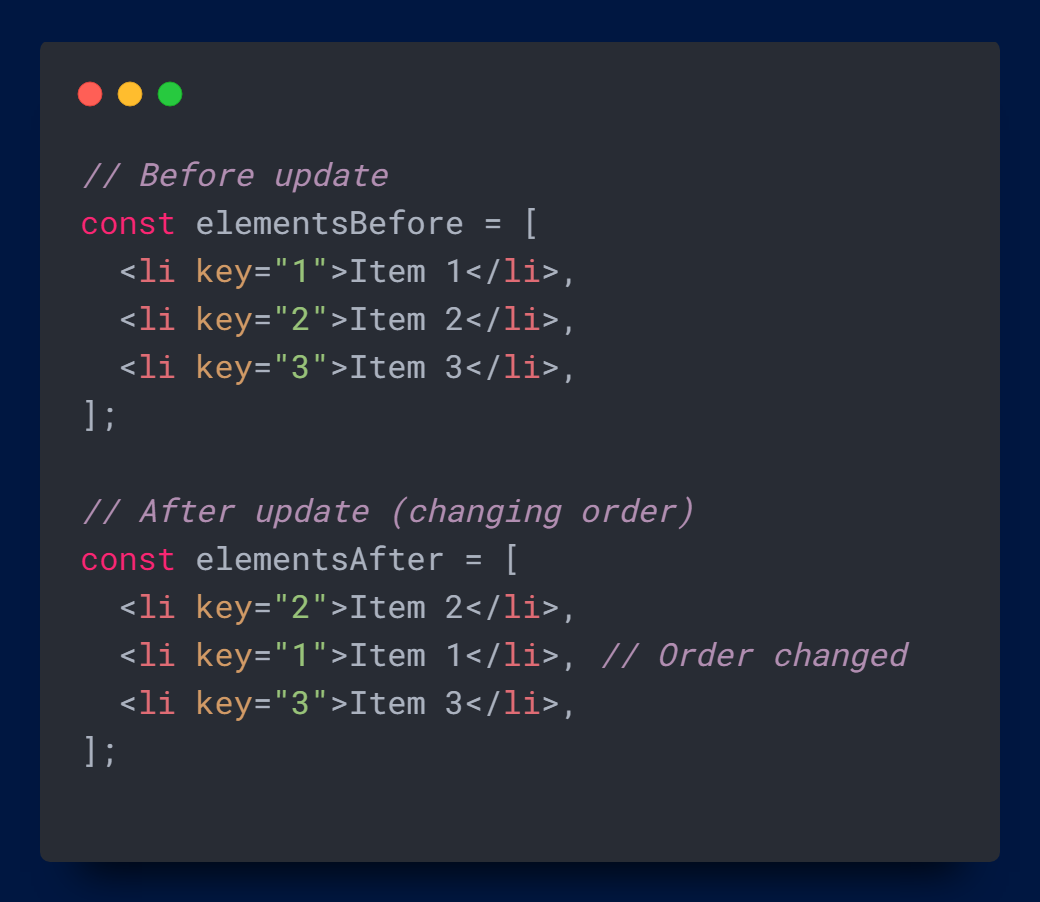
In React, when comparing two elements of different types during the reconciliation process, React assumes that the content of the elements has completely changed, and it treats them as completely different subtrees. This is a fundamental principle in React's diffing algorithm and is often referred to as the "reconciliation assumption."



In this example, elementBefore and elementAfter have different root types (div vs. span). When React reconciles these elements, it will consider the entire subtree to be different, even though the only change is the root element type. React will unmount the old div and mount the new span, resulting in the DOM being completely replaced.

This approach helps React ensure that it doesn't make assumptions about how different types of elements should be updated. Treating them as completely different subtrees provides a clear and predictable behavior, even if it may seem less efficient for small changes in the structure of the elements.

**ELEMENTS WITH A STABLE KEY PROP STAY THE SAME ACROSS THE RENDERS**

****

When elements have a stable and unique key prop, React uses the key to optimize the update process during reconciliation. The key prop helps React identify which elements in the new set correspond to which elements in the previous set, even when the order of elements changes. When React finds elements with matching keys, it assumes that the elements represent the same underlying data and attempts to update them rather than unmounting and remounting.

In this example, each li element has a unique and stable key prop. When React reconciles the before and after sets of elements, it uses the key prop to determine that the first and second elements have swapped places. Instead of unmounting and remounting the elements, React updates their positions efficiently.

Using stable keys is important when dealing with dynamic lists of elements. It helps React correctly identify and update elements as the order changes, leading to more efficient rendering and a better user experience.

**Do you find it helpful?**

**Let me know down in the comments**



[](https://www.linkedin.com/in/anubhav-gupta-b68946132/)

**Click To Follow For More On LinkedIN**