**How React Implements useState() Hook**

Let's break down the **useState()** hook in a simplified manner to understand its inner workings.

At a high level, React needs to:

1. Keep track of the state value.
2. Ensure this state is persistent across renders.
3. Allow updates to this state.
4. Trigger a re-render when the state changes.

**Basic Concept**:

Every component instance that uses **useState()** has an associated list (or array) that keeps track of all the states for that component.

When you call **useState()**, React does the following:

1. Checks if this is the first time the component is rendering.
2. If it's the first render, initializes the state with the value you provided.
3. Adds this initial state value to the component's state list.
4. If it's a subsequent render (due to an update), React retrieves the current state value from the component's state list based on the order of **useState** calls.
5. Provides a function (**setState**) that you can use to update this state. When called, it updates the value in the state list and triggers a re-render of the component.

**Pseudo-Code Implementation:**

let globalStateArray = [];

let globalIndex = 0;

function useState(initialValue) {

const currentIndex = globalIndex;

globalStateArray[currentIndex] = globalStateArray[currentIndex] || initialValue;

function setState(newValue) {

globalStateArray[currentIndex] = newValue;

renderComponent(); // Pseudo-code to trigger a re-render

}

globalIndex++;

return [globalStateArray[currentIndex], setState];

}

**Things to Keep in Mind:**

1. **Order Matters**: The order of **useState** (and other hooks) calls must be consistent between renders. This is because React relies on the call order to match each state with its updater function. This is also why hooks cannot be called conditionally (inside loops, conditions, or nested functions).
2. **Real Implementation**: The actual implementation in React is more complex, given that React handles numerous edge cases, optimizations, concurrent mode, etc. The provided pseudo-code is a basic representation for illustrative purposes.
3. **Multiple States**: Each component can have multiple states. The array-based approach (with the order of calls) is a simplified way of how React might keep track of multiple state values and updater functions.

In conclusion, **useState** provides a mechanism to store, retrieve, and update state values across component renders. It leverages the internal workings of React to ensure consistent and performant behavior.