In React, both **props** and **state** are used to manage and handle data, but they serve different roles and have different characteristics.

**Props (short for "properties")**

1. **Immutable**: Props are read-only and should not be modified within the component that receives them. If a parent component passes a prop to a child component, the child should only use it, not modify it.
2. **Parent to Child**: Props allow data to be passed from a parent component to a child component.
3. **Functional Role**: Often used to pass data and event handlers down to child components.
4. **Initial Values**: Often used to provide initial values for the state variables in the child component.
5. **Static or Dynamic**: The value of a prop can be a static value set by the parent or dynamically generated based on some computation or state.
6. **Data Types**: Props can be of any data type: numbers, strings, arrays, objects, functions, etc.

**State**

1. **Mutable**: State is mutable, meaning it can be changed within the component using the **setState** function (or the **useState** hook in functional components).
2. **Local or Component Scope**: State exists within the component and controls the behavior of the component itself, but it can also be lifted up to a parent component to be shared with children.
3. **Dynamic Role**: Often used for dynamic data that may change over time due to user interactions, network requests, etc.
4. **Trigger Re-render**: Changing the state will trigger a re-render of the component to reflect the new state.
5. **Initialization**: Usually initialized in the constructor (in class components) or by using the **useState** hook (in functional components).
6. **Data Types**: State can be of any data type: numbers, strings, arrays, objects, etc.

**Examples**

Props

jsxCopy code

// ParentComponent.jsx const ParentComponent = () => { return <ChildComponent name="John" />; }; // ChildComponent.jsx const ChildComponent = (props) => { return <div>Hello, {props.name}!</div>; };

State

jsxCopy code

// CounterComponent.jsx import React, { useState } from 'react'; const CounterComponent = () => { const [count, setCount] = useState(0); return ( <div> <button onClick={() => setCount(count + 1)}>Increment</button> <p>Count: {count}</p> </div> ); };

In summary, **props** are used to pass data from parent to child components and are immutable, while **state** is used to manage dynamic data within a component and can be changed.