Difference Between State, Props, and Hooks in React

1. State

Definition: State represents the dynamic data or values that change over time within a component.

It is managed within the component itself and can be updated based on user interactions or other events.

Usage: State is used to control and store data that may change within the component.

Example:

```
const [count, setCount] = useState(0); // useState Hook to manage state
```

Key Point: State is mutable (can be updated) and is specific to the component where it is defined.

2. Props

Definition: Props (short for "properties") are read-only attributes passed from a parent component to a child component.

They are used to pass data between components.

Usage: Props allow components to receive data from a parent component and use it to render or perform actions.

Example:

```
function Header({ title }) {
   return <h1>{title}</h1>; // Accessing the prop 'title'
```

}

Key Point: Props are immutable (cannot be changed by the child component) and are passed down from a parent to a child component.

3. Hooks

Definition: Hooks are functions introduced in React 16.8 that allow you to "hook into" React features (like state and lifecycle methods) within functional components.

Usage: Hooks manage state, side effects, and context in functional components, providing features that were previously only available in class components.

Example:

```
const [count, setCount] = useState(0); // useState Hook
useEffect(() => {
  // Runs side effect (e.g., fetch data)
}, []);
```

Key Point: Hooks allow you to use state, side effects, and other React features in functional components.

Summary:

- State is internal to a component and can change over time.
- Props are external data passed from a parent component and are immutable.
- Hooks are functions that manage logic like state and side effects within functional components, enabling more flexible and reusable code.