## Hooks

Note : https://dev.to/johnstonlogan/react-hooks-barney-style-1hk7#:~:text=setState%20%28%29%20vs%20useState%20%28%29%20-%20Objects.%20setState,to%20populate%20the%20values%20of%20the%20state%20object.

Hooks were added to React in version 16.8.

Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.

Note : In React, a state variable is a variable that holds the state of a component. It is used to store data that can be changed over time and can be used to re-render the component when the data changes. [The useState hook is used to add state to functional components in React](https://reactjs.org/docs/hooks-state.html" \t "https://www.bing.com/_blank)

In case of class component, when we use state then in that case we need to write this.state = {name:”Andrew”}; but in case of functional component we need to write const[name,setValue]=useState(“”);

So here the name is internally considered as a state like this.state = {name:”Andrew”}; because we declare state in class component in this way. Or we can also call it as a state variable.

Although Hooks generally replace class components, there are no plans to remove classes from React.

Note :

What is State in React? State is a****JavaScript object**** that stores component’s dynamic data

React useState is the React Hook that allows you to manage the state within functional components.

**What is the usestate function?**

The useState function is a built in hook that can be imported from the react package. It allows you to add state to your functional components. Using the useState hook inside a function component, you can create a piece of state without switching to class components.

## Built-in Hooks

Here, we describe the APIs for the built-in Hooks in React. The built-in Hooks can be divided into two parts, which are given below.

****Basic Hooks****

* useState
* useEffect
* useContext

****Additional Hooks****

* useReducer
* useCallback
* useMemo
* useRef
* useImperativeHandle
* useLayoutEffect
* useDebugValue

useRef

The useRef Hook allows you to persist(दृढ़ रहना) values between renders.

* It can be used to access a DOM element directly.

**Does Not Cause Re-renders**

* If we tried to count how many times our application renders using the useState Hook, we would be caught in an infinite loop since this Hook itself causes a re-render.
* To avoid this, we can use the useRef Hook.

**Accessing DOM Elements**

* In general, we want to let React handle all DOM manipulation.
* But there are some instances where useRef can be used without causing issues.
* In React, we can add a ref attribute to an element to access it directly in the DOM.