State:

- ✓ The state is a built-in React object that is used to contain data or information about the component. State could only be used in class components.
- ✓ State allows us to manage changing data in an application.
- ✓ To define a state, add a class constructor which assigns an initial state using this.state.

```
import React, { Component } from 'react'

export class Student extends Component {

 //let city="Pune";     //Unexpected token. A constructor, method or property was expected.

 constructor(){
   super();              //'this' is not allowed before 'super()'
   this.state={
        name:"Sachin",
        course:"Java"
   } }
 render() {
   return (<>
      <h1>Name : {this.state.name} and Selected Course : {this.state.course}</h1>
   </>)
 }
}
```

Props:

- ✓ Props stands for properties. Props are like function arguments in JavaScript.
- ✓ Props are arguments passed into React Components. (from Parent to Child only).
- ✓ Components can pass information to other components. When one component passes information to another, it is passed as ***props*** through one or more ***attributes***.
- ✓ Props are immutable so we cannot modify the props from inside the component.
- ✓ To send props into a component, use the same syntax as HTML attributes.

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<>
      <div className='container bg-light'>
         <h1 className='text-primary'>Welcome to React App</h1>
         <Laptop brand="HP"/>
      </div>
 </>);
```

The Component receives the argument as a Props Object in Laptop.js

```
export function Laptop(props) {
  return <h1 className="text-secondary">This is {props.brand} Laptop</h1>
}
```

- ✓ create a variable and send it to the laptop component:

```
const site="https://www.hp.com/";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<>
       <div className='container bg-light'>
          <h1 className='text-primary'>Welcome to React App</h1>
          <Laptop brand="HP" website={site}/>
       </div>
 </>);
```

Laptop.js

```
export function Laptop(props) {
  return (<>
     <h1 className="text-success">This is {props.brand} Laptop</h1>
     <h2 className="text-secondary">Connect with Us On : {props.website}</h2>
  </>)
}
```

React Context:

- ✓ React Context allows us to easily access data at different levels of the component tree without passing props.
- ✓ We need to create context first by using createContext()

Parent.js

```
import { createContext } from "react";
import AComp from "./AComp";

export const Username=createContext();
export const Password=createContext();

const Parent=()=>{
    return (<>
          <Username.Provider value="Root">
             <Password.Provider value="root@123#">
                <AComp/>
             </Password.Provider>
          </Username.Provider>
    </>)
}
export default Parent;
```

Dcomp.js

```
import React from 'react'
import { Password, Username } from './Parent'
```

```
function DComp() {
  return (
    <>
      <h1>This is D Component!</h1>
      <Username.Consumer>
        {   tmp => {
          return (<>
          <Password.Consumer>
           {
            function (psw) {
               return (<>
                          <h1>Username : {tmp} & Password : {psw}</h1>
                  </>)   }
            }
           </Password.Consumer>
          </>)
        }}
      </Username.Consumer>
    </>
  )
}
export default DComp
```

With useContext()Hook:
Dcomp.js

```
import React, { useContext } from 'react'
import { Password, Username } from './Parent'

function DComp() {
  const user = useContext(Username);
  const pass = useContext(Password);
  return (
    <>
      <h1>This is D Component!</h1>
      <h1>Username : {user} & Password : {pass}</h1>
    </>
  )
}
export default DComp
```

React Hooks:

- ✓ Hooks are the new feature introduced in the React 16.8 version (2019).
- ✓ It allows you to use state and other React features without writing a class. It does not work inside classes. Because of this, class components are generally no longer needed.
- ✓ Hooks allow function components to have access to state and other React features.

Rules Of Hooks:

1. Hook can only be called inside Functional components.
2. Make sure to not use Hooks inside loops, conditions, or nested function.
3. Hooks cannot be conditional.

useState():

- ✓ The React useState Hook allows us to track state in a function component and use to change the state of an object.
- ✓ UseState accepts an initalState & return two values:
  - const [state,setState]=useState(initialState)
- ✓ First value is current State & second value is setState, which is the function that is used to update our state.

Counter.js

```
import React, { useState } from 'react'

export default function Counter() {


   const [count,setCount]=useState(0)

  return (
    <>
      <h1 className='text-secondary'>Count :{count} </h1>
      <button onClick={()=>setCount(0)} className="btn btn-outline-primary m-
2">Reset</button>

      <button onClick={()=>setCount(count + 1)}  className="btn btn-outline-success m-
2">Increment</button>

      <button onClick={()=>setCount(count - 1)}  className="btn btn-outline-danger m-
2">Decrement</button>
    </>
  )
}
```

useEffect():

- ✓ The useEffect Hook allows you to perform side effects in your components.
- ✓ Examples of side-effects are fetch requests(API), manipulating DOM directly and more.
- ✓ useEffect() hook accepts 2 arguments: useEffect(callback,[dependencies]);
- ✓ callback is the function containing the side-effect logic. callback is executed right after changes were being pushed to DOM.
- ✓ dependencies is an optional array of dependencies. useEffect() executes callback only if the dependencies have changed between renderings.
- ✓ Empty dependencies array[] will make useEffect to run only once at startup because that array never change.

Counter.js

```
import React, { useEffect, useState } from 'react'

export function CounterHook() {
   const [count,setCount]=useState(0)

   useEffect(()=>{
      console.log("useEffect Call!")
   })    //,[] , count===5]

  return (<>
         <h1>Count : {count}</h1>
         <button onClick={()=>setCount(count + 1)}
          className="btn btn-outline-success">Click Here</button>
  </>)
}
```