

Get familiar
with **5** essential

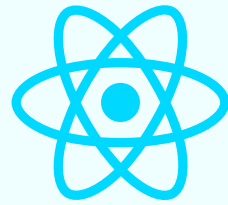
React Hooks

that are frequently used.



Nahidul Islam

<https://nahidul-islam-fahim.web.app>



→ useState

useState is a hook used for adding state variables to functional components. It returns an array with the current state value and a function to update that state.

```
useState Hook

import { useState } from 'react';

const Counter = () => {
  const [count, setCount] = useState(0);

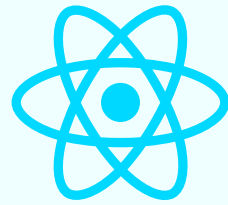
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
};
```



Nahidul Islam

<https://nahidul-islam-fahim.web.app>

1



→ useEffect

useEffect is used for handling side effects in functional components. It runs after every render and is commonly used for data fetching, subscriptions, or manual DOM manipulations.

```
useEffect Hook

import { useEffect, useState } from 'react';

const DataFetcher = () => {
  const [data, setData] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      const result = await fetchDataFromAPI();
      setData(result);
    };

    fetchData();
  }, []); // Empty dependency array means it runs once after the initial render.

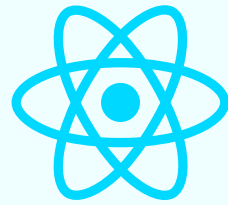
  return <p>Data: {data}</p>;
};
```



Nahidul Islam

<https://nahidul-islam-fahim.web.app>

2



→ useContext

useContext is used for consuming the value of a React context within a functional component, allowing us to avoid prop drilling.

```
useContext Hook

import { createContext, useContext } from 'react';

const MyContext = createContext();

const ParentComponent = () => (
  <MyContext.Provider value="Hello from Context">
    <ChildComponent />
  </MyContext.Provider>
);

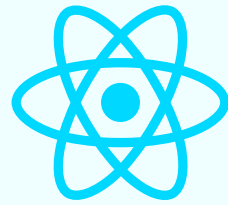
const ChildComponent = () => {
  const contextValue = useContext(MyContext);
  return <p>{contextValue}</p>;
};
```



Nahidul Islam

<https://nahidul-islam-fahim.web.app>

3



→ useRef

useRef is primarily used for accessing and interacting with the DOM directly, but it can also be used to persist values across renders without causing re-renders.

```
useRef Hook

import { useRef, useEffect } from 'react';

const InputWithFocus = () => {
  const inputRef = useRef(null);

  useEffect(() => {
    // Focus on the input element after the component mounts
    inputRef.current.focus();
  }, []);

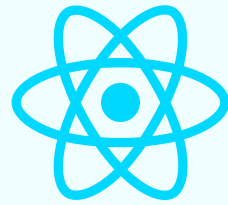
  return <input ref={inputRef} />;
};
```



Nahidul Islam

<https://nahidul-islam-fahim.web.app>

4



→ useMemo

useMemo is used to memoize the result of a computation, preventing unnecessary recalculations. It's particularly useful for optimizing performance in scenarios involving expensive operations.

```
useMemo Hook

import { useMemo } from 'react';

const ExpensiveComponent = ({ a, b }) => {
  const result = useMemo(() => {
    // Expensive computation based on 'a' and 'b'
    return a * b;
  }, [a, b]);

  return <p>Result: {result}</p>;
};
```



Nahidul Islam

<https://nahidul-islam-fahim.web.app>

5

These hooks showcase essential React concepts, but there are many more with varying functionalities to discover.

Thank you for your time!



Nahidul Islam

<https://nahidul-islam-fahim.web.app>

<https://github.com/nahidul-fahim>

