|  |
| --- |
| **Component Mounting:**  import React, { useState, useEffect } from 'react';  function HomeComponent() {  useEffect(() => {  // This code will run after the component is mounted.  console.log('Component is mounted.');  }, [ ]);  return (  <div>  <p>Home Component</p>  </div>  );  } |
| **Component Updating:**  import React, { useState, useEffect } from 'react';  function AboutComponent() {  const [count, setCount] = useState(0);  useEffect(() => {    // This code will run whenever `count` changes.  console.log(`Count is now: ${count}`);  }, [count]);  return (  <div>  <p>Count: {count}</p>  <button onClick={() => setCount(count + 1)}>Increment</button>  </div>  );  } |
| **Component Unmounting (cleanup):**  import React, { useState, useEffect } from 'react';  function ContactComponent() {  const [visible, setVisible] = useState(true);  useEffect(() => {  // This code runs when the component is mounted.  console.log('Component is mounted.');  // This cleanup function runs when the component is unmounted.  return () => {  console.log('Component will unmount.');  };  }, [ ]);  return (  <div>  <p>Example Component</p>  <button onClick={() => setVisible(!visible)}>Toggle Component</button>  {visible && <p>Visible Content</p>}  </div>  );  } |
| **Data Fetching:**  import React, { useState, useEffect } from 'react';  function DataFetchingComponent() {  const [data, setData] = useState(null);  useEffect(() => {  // This code runs when the component is mounted.  fetch('https://api.example.com/data')  .then((response) => response.json())  .then((result) => {  setData(result);  });  }, [ ]);  return (  <div>  {data ? (  <p>Data: {data}</p>  ) : (  <p>Loading data...</p>  )}  </div>  );  } |
| **Cleaning :**  import React, { useState, useEffect } from 'react';  function ComponentWithCleanup() {  const [count, setCount] = useState(0);  useEffect(() => {  // This effect subscribes to some event or resource when the component mounts  const subscription = subscribeToEvent();  // This is the cleanup function  return () => {  // Unsubscribe or clear the resource when the component unmounts  subscription.unsubscribe();  };  }, []); // Empty dependency array means this effect runs only once on mount  const subscribeToEvent = () => {  // Simulate subscribing to some event (e.g., a WebSocket, Redux store, etc.)  const eventSubscription = {  unsubscribe: () => {  console.log('Unsubscribed from the event');  },  };  return eventSubscription;  };  return (  <div>  <p>Count: {count}</p>  <button onClick={() => setCount(count + 1)}>Increment</button>  </div>  );  }  export default ComponentWithCleanup; |