React Hooks

useCallback,useEffect, and useRef

hardik agnihotri

useCallback

When to Use useCallback

- **Prevent Unnecessary Re-Renders: When you pass a function as a prop to a child component, useCallback ensures the function reference remains stable unless its dependencies change, avoiding re-renders.**
- **Expensive Calculations: If a function performs complex calculations and is used as a prop or within useEffect, memoizing it can enhance performance.**

Example

hardik agnihotri

Example

```jsx
import React, { useState, useCallback } from 'react';

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

  const increment = useCallback(() => {
    setCount((prevCount) => prevCount + 1);
  }, []); // Empty dependency array ensures this function is only created once

  return (
    <div>
      <button onClick={increment}>Increment</button>
      <p>Count: {count}</p>
    </div>
  );
}

export default Counter;
```

In this example, the increment function is memoized using useCallback, ensuring that its reference remains the same across renders.

useEffect >

hardik agnihotri

useEffect

When to Use useEffect

- Side Effects: Operations like data fetching, subscriptions, or manually changing the DOM are considered side effects.
- Dependency Array: Controls when the effect runs. An empty array [ ] runs the effect only once, while a populated array [dependency1, dependency2] runs the effect whenever specified dependencies change.

Example

hardik agnihotri

Example

```jsx
function FetchData() {
  const [data, setData] = useState([]);

  useEffect(() => {
    async function fetchData() {
      const response = await fetch('https://api.example.com/data');
      const result = await response.json();
      setData(result);
    }

    fetchData();
  }, []); // Effect runs only once, similar to componentDidMount

  return (
    <ul>
      {data.map((item) => (
        <li key={item.id}>{item.name}</li>
      ))}
    </ul>
  );
}
```

Here, useEffect fetches data once when the component mounts, thanks to the empty dependency array.

useRef >

hardik agnihotri

useRef

When to Use useRef

- Accessing DOM Elements: Use useRef to interact with DOM elements without causing additional renders.
- Persisting Values: Store any mutable value that should persist between renders without causing the component to re-render.

Example

hardik agnihotri

Example

```jsx
import React, { useRef } from 'react';

function FocusInput() {
  const inputRef = useRef(null);

  const focusInput = () => {
    if (inputRef.current) {
      inputRef.current.focus();
    }
  };

  return (
    <div>
      <input ref={inputRef} type="text" />
      <button onClick={focusInput}>Focus Input</button>
    </div>
  );
}

export default FocusInput;
```

In this example, useRef is used to store a reference to the input element, allowing us to focus it when the button is clicked.

End

hardik agnihotri

DID YOU FIND IT HELPFUL ?

Follow for more

hardik agnihotri

Share this with a friend who needs it!