React Hooks Guide with Examples

1. useState

useState is the hook that allows you to add state to a functional component.

It returns a state variable and a function to update that state.

Example 1:

```jsx
import { useState } from 'react';
function Counter() {
    const [count, setCount] = useState(0);
    return (
        <div>
            <p>Count: {count}</p>
            <button onClick={() => setCount(count + 1)}>Increase</button>
        </div>
    );
}
```

Example 2:

```jsx
import { useState } from 'react';
function TextInput() {
    const [text, setText] = useState("");
    return (
        <div>
            <input type="text" value={text} onChange={(e) => setText(e.target.value)} />
            <p>Text: {text}</p>
        </div>
    );
}
```

useState allows the component to have its own state that can change over time.

2. useEffect

useEffect is the hook that runs side effects in your component.

It is used for actions like fetching data, subscribing to external data sources, or manually changing the DOM.

Example 1:

```jsx
import { useState, useEffect } from 'react';
function DataFetcher() {
    const [data, setData] = useState(null);

    useEffect(() => {
        fetch('https://api.example.com/data')
            .then(response => response.json())
            .then(data => setData(data));
    }, []);  // Empty array means it runs only once (on component mount)
    return <div>{data ? JSON.stringify(data) : "Loading..."}</div>;
}
```

Example 2:

```jsx
import { useState, useEffect } from 'react';
function Timer() {
    const [time, setTime] = useState(0);

    useEffect(() => {
        const timer = setInterval(() => setTime(time => time + 1), 1000);
        return () => clearInterval(timer);  // Cleanup on unmount
    }, []);  // Runs only once on mount

    return <div>Time: {time}s</div>;
}
```

useEffect is commonly used for data fetching, timers, or changing the DOM manually.