Nombre: José Ortega
Ficha: 3064781

Actividad: Construir un pequeño “contador dinámico” con React y React Hooks (useState)

Objetivo: entender el uso básico de useState, cómo declarar una variable de estado, cómo actualizarla, cómo reflejarla en el UI.

Paso a paso:

1. Crea un nuevo componente funcional en React (puede ser Counter.jsx).
2. Importa useState:
3. import React, { useState } from 'react';
4. Dentro del componente, declara una variable de estado count (o “contador”) inicializada en 0:
5. const [count, setCount] = useState(0);
6. En el return, renderiza un párrafo que muestre el valor del contador, por ejemplo:
7. Has hecho clic {count} veces
8. Añade un botón que, al hacer clic, llame a setCount(count + 1). Ejemplo:
9. <button onClick={() => setCount(count + 1)}>Incrementar</button>
10. Extensión opcional: añade otro botón para “Reiniciar” el contador (reset a 0), e incluso otro para “Decrementar” (si quieres).
11. (Si te sientes aventurero) Modifica para que, en lugar de simplemente incrementar en 1, puedas especificar un valor en un input y actualizar el contador con ese valor, usando otro estado para almacenar ese “valor de entrada”.

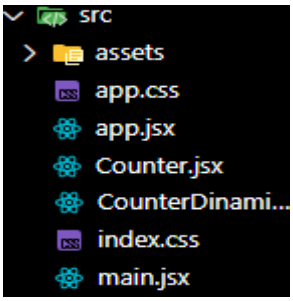
Qué observar / qué preguntas responder:

- ¿Qué pasa cuando haces clic muchas veces? ¿El valor cambia como esperas?
- ¿Qué sucede si pasas a useState un valor que no sea 0, por ejemplo, useState(10)?
- ¿Por qué llamamos a la función setCount para cambiar el estado? ¿Qué pasa si tratas de modificar count directamente?
- Si implementas el input para valor dinámico, ¿cómo haces que ese valor se “escuche” y se use para actualizar el contador?
- Reflexiona: ¿por qué useState sólo se llama al nivel del componente y no dentro de bucles o condiciones? (la documentación lo señala) es.react.dev

Nombre: José Ortega
Ficha: 3064781

Solución

Archivos creados Counter.jsx y CounterDinamico.jsx



app.jsx

```jsx
import { useState } from 'react';
import Counter from './Counter.jsx'
import CounterDinamico from './CounterDinamico.jsx'

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

  const [countValor, setCountValor] = useState(0);
  const [valor, setValor] = useState(0);
  return (
    <div>
      <Counter conteo={count} establecer={setCount}></Counter>
      <CounterDinamico valor={valor} setValor={setValor} countValor={countValor} setCountValor={setCountValor}></CounterDinamico>

    </div>
  )
}
```

Counter.jsx

```jsx
function Counter({conteo, establecer}) {
    return (
        <div>
            <p>Has hecho clic {conteo} veces</p>
            <button onClick={() => establecer(conteo + 1)}>Incrementar</button>
            <button onClick={() => establecer(conteo - 1)}>Decrementar</button>
            <hr />
        </div>
    );
}
export default Counter;
```

CounterDinamico.jsx

```jsx
function CounterDinamico({valor, setValor, countValor, setCountValor}) {
    return (
        <div>
            <p>Valor dinamico {valor}</p>
            <p>Has hecho clic {countValor} veces</p>
            <input type="number" value={valor} onChange={(e) => setValor(Number(e.target.value))}/>
            <button onClick={() => setCountValor(countValor + valor)}>Incrementar por valor</button>
            <button onClick={() => setCountValor(countValor - valor)}>Decrementar por valor</button>
            <button onClick={() => setCountValor(0)}>Reiniciar</button>
            <hr />
        </div>
    );
}
export default CounterDinamico;
```

Nombre: José Ortega
Ficha: 3064781

1. ¿Qué pasa cuando haces clic muchas veces? ¿El valor cambia como esperas?
   - El valor del contador cambia correctamente al presionar los botones, aumentando o disminuyendo según el botón que se use.

2. ¿Qué sucede si pasas a useState un valor que no sea 0, por ejemplo, useState(10)?
   - Si se cambia el valor inicial de useState, por ejemplo a 10, el contador empezará desde ese número cuando se cargue el componente.

3. ¿Por qué llamamos a la función setCount para cambiar el estado? ¿Qué pasa si tratas de modificar count directamente?
   - Porque setCount es la única forma segura de decirle a React que el estado cambió. Si se intenta modificar count directamente (por ejemplo: count = count + 1), el valor sí cambia en la variable, pero React no se entera del cambio, así que no vuelve a renderizar el componente y la interfaz no se actualiza. En cambio, al usar setCount, React actualiza el valor y vuelve a dibujar el componente con el nuevo estado.

4. Si implementas el input para valor dinámico, ¿cómo haces que ese valor se "escuche" y se use para actualizar el contador?

```
const [countValor, setCountValor] = useState(0);
const [valor, setValor] = useState(0);
```

```
<div>
    <p>Valor dinamico {valor}</p>
    <p>Has hecho clic {countValor} veces</p>
    <input type="number" value={valor} onChange={(e) => setValor(Number(e.target.value))}/>
    <button onClick={() => setCountValor(countValor + valor)}>Incrementar por valor</button>
    <button onClick={() => setCountValor(countValor - valor)}>Decrementar por valor</button>
</div>
```

5. Reflexiona: ¿por qué useState sólo se llama al nivel del componente y no dentro de bucles o condiciones? (la documentación lo señala) es.react.dev
   - Porque React necesita saber el orden exacto en que se llaman los Hooks en cada renderizado. Si los colocas dentro de un if, for o función condicional, ese orden podría cambiar, y React no sabría qué estado corresponde a cada Hook, causando errores. Por eso, los Hooks deben estar siempre al nivel superior del componente, nunca dentro de condicionales o bucles.