Definiciones

Hooks: los hooks son funciones especiales que te permiten "engancharte" a características de React, como el estado y el ciclo de vida de los componentes, sin tener que escribir una clase. Permite que los componentes funcionales sean más poderosos y versátiles.

El hook useState es uno de los hooks más utilizados en React y se utiliza para manejar el estado en componentes funcionales.

El hook useEffect se utiliza para manejar efectos secundarios en componentes funcionales. Por ejemplo para hacer solicitudes a APIs, suscribirse a eventos, modificar el DOM directamente, entre otros fines

Ver documentación.

Explicación adicional.

Manejo de solicitudes (en el backend):

```
// src/models/index.js


const all = (req, res) => {
    const sql = 'SELECT * FROM curso';
    db.all(sql, [], (err, rows) => {
        if (err) {
            res.status(400).json({ error: err.message });
            return;
        }
        res.json({ data: rows });
    });
};
```

req representa la solicitud HTTP entrante.

res representa la respuesta que se enviará al cliente.

Cuando el cliente (por ejemplo, una aplicación React) hace una solicitud a un endpoint correspondiente (ej: /), esta función se invoca.

Gestión de datos (en el frontend):

// src/components/Form.jsx

function Form() { ver codigo en el ejemplo }

Ante el código del backend que envía datos al frontend, les defino a continuación, el flujo de datos en React.

Flujo de datos en React

1. Backend

- El servidor está corriendo en el localhost y tiene dos endpoints (get y post, por ejemplo).
- Cuando el componente se monta, hace una solicitud GET a este endpoint para obtener la lista de cursos.

2. Uso de useEffect

- El hook eseEffect se ejecuta una vez al montar el componente (debido al array vacío `[]`).
- En este hook, se realiza una llamada fetch al servidor: `fetch('http://localhost:3001/')`
- Una vez que el servidor responde, el código convierte la respuesta a JSON: `response.json()`
- Se espera que la estructura del JSON devuelto tenga una clave data que contenga la lista de cursos: `.then(data => setCursos(data.data))`
- Aquí se actualiza el estado cursos con los datos obtenidos.
- Nota: la estructura de data es { "data": [{ "id_curso": 1, "titulo": "Curso 1" }, { "id_curso": 2, "titulo": "Curso 2" }] }. Entonces, data.data representa la manipulación del objeto JSON interno, que contiene el array de cursos.

3. Renderizado de la Lista de Cursos

- El estado cursos es un array que se mapea en el return del componente: `{cursos.map((curso) => ( <li key={curso.id_curso}>{curso.titulo}</li> ))}`
- Por cada curso, se crea un elemento de lista `<li>` que muestra el título del curso.