educación IT
CODE YOUR FUTURE

ReactJS Developer

Módulo 3

educación IT
CODE YOUR FUTURE

Uplifting

Uplifting

Usualmente muchos componentes necesitan **reflejar el mismo cambio en datos compartidos.** Pensemos, por ejemplo, en el cambio de idioma o el cambio del tema de la aplicación. Para eso vamos a usar una estrategia denominada **“Levantar el estado”** o **“*Uplifting*”**.

Esta estrategia consiste en **definir una “única fuente de verdad”** lo más arriba posible del árbol de componentes.

En otras palabras

- Una única fuente de verdad significa que **el estado compartido debe estar almacenado en un solo componente** (en lugar de repetirse el mismo dato en varios componentes)
- Lo más arriba posible del árbol hace referencia al sentido unidireccional de los cambios de estado: los cambios de estado ocurren de padres a hijos (y no al revés). Para que dos componentes respondan al mismo cambio, **ese cambio debe ser hecho en un componente padre que los englobe.**

Implementación

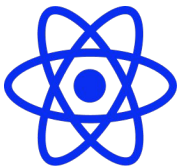
Para implementar esta estrategia necesitamos gestionar dos comunicaciones: **pasar el estado del padre al hijo y notificar cambios de estado del hijo al padre.**

Consideraciones

- Recordemos que el estado compartido **debe estar almacenado en el padre.**
- Pasar el estado del padre al hijo es muy sencillo y se hace **a través de las `props`.**

Las **`props`** son una excelente forma de pasar un dato de un componente padre a otro hijo. Sin embargo, pasar el estado del hijo al padre no es tan sencillo pues no puede hacerse directamente. Para lograr eso, **debemos crear nuestros propios eventos.** Podemos recibir en las `props` funciones **cuyo cuerpo esté en el padre y cuya llamada esté en el componente actual.**

Dicho con otras palabras: este paso consiste **en crear una función en el padre (con acceso a su estado), pasarla a través de las props al hijo, y ejecutarla en el hijo cuando sea necesario.** Recuerda que lo que estás usando en el hijo, es una función, por lo que puedes pasarle tantos parámetros como quieras.



Ejemplo

Aquí tienes un ejemplo completo basado en una calculadora de conversión de divisas.

```jsx
import CurrencyField from "./CurrencyField";
import { useState } from "react";

function Calculator(props) {
    const [value, setValue] = useState(0);
    const [divisa, setDivisa] = useState("");

    const handleUsdChange = (val) => {
        setDivisa("usd");
        setValue(val);
    };

    const handleArsChange = (val) => {
        setDivisa("ars");
        setValue(val);
    };

    const usdValue = divisa === "ars" ? value / 200 : value;
    const arsValue = divisa === "usd" ? value * 200 : value;

    return (
        <div>
            <CurrencyField
                divisa="usd"
                value={usdValue}
                onChange={handleUsdChange}
            />
            <CurrencyField
                divisa="ars"
                value={arsValue}
                onChange={handleArsChange}
            />
        </div>
    );
}

export default Calculator;
```

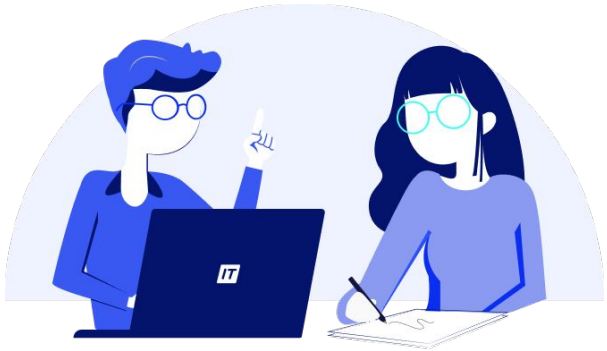
```
function CurrencyField(props) {
    return (
        <fieldset>
            <legend>{props.divisa.toUpperCase()}</legend>
            <input
                type="number"
                value={props.value}
                onChange={(e) => props.onChange(e.target.value)}
            />
        </fieldset>
    );
}

export default CurrencyField;

```

Consideraciones finales

Si bien esta estrategia es muy útil, experimentarás que **no sirve cuando debemos compartir datos a lo largo de muchos elementos del árbol** o de forma **muy profunda** (en vez de padres a hijos, de abuelos a nietos o más).

En estos casos, esta estrategia no es recomendable. En su lugar, se opta por *Context API* o el uso de *Redux*.

Sin embargo, para pequeñas comunicaciones es una estrategia **sumamente válida.**



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!