Nivel 4: Props y Estado en React

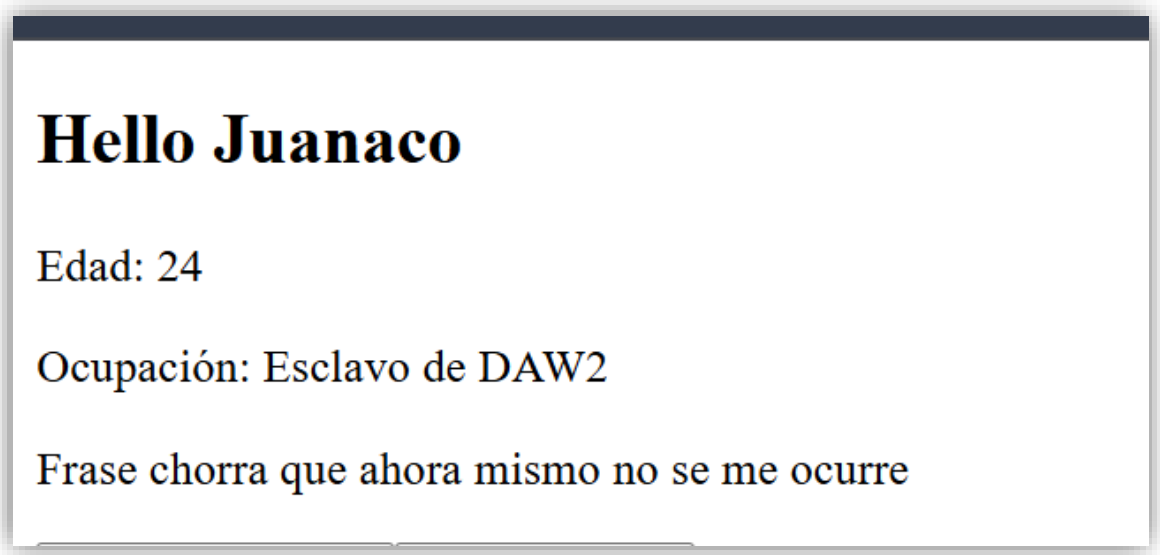
Tabla de contenido

1. Paso de parámetros: Props (Parte A, B y C)

He configurado un componente llamado ParentComponent que actúa como contenedor principal. Desde aquí, he pasado diferentes tipos de datos al componente llamado ChildComponent.

Para demostrar que domino el paso de props, no me he limitado a textos simples; he pasado:

- **Un string con mi nombre.**
- **Un number para mi edad.**
- **Un array con mis hobbies.**
- **Una función de saludo personalizada.**



2. El concepto de Inmutabilidad (Parte D)

En esta práctica he observado que un componente hijo no puede modificar directamente las props que recibe. El valor name llega al componente hijo como prop y no se cambia directamente, sino que se actualiza mediante la función setName, que pertenece al componente padre.

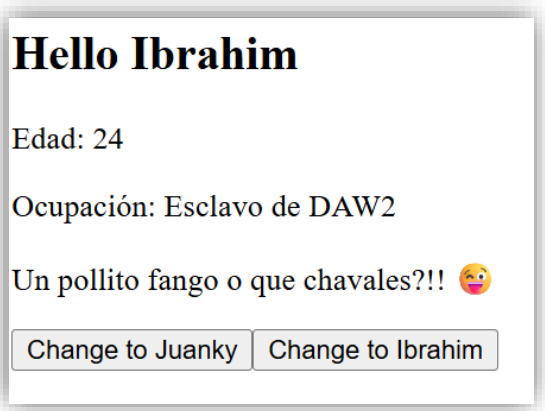
Esto demuestra que las props son inmutables y que los cambios de datos deben realizarse siempre a través del estado del componente padre.

```
function ChildComponent(props) {
    props.name = "Pablo" // NO se permite
    return <p>Hello profesora (guiño guiño)! my name is
    {props.name}</p>    'name' is missing in props validation
}

export default ChildComponent
```

3. Gestión del Estado y Comunicación Inversa (Parte E y F)

Para hacer que la aplicación sea interactiva, he utilizado un estado con useState. Lo más interesante ha sido pasar el estado y la función setName al hijo como props.

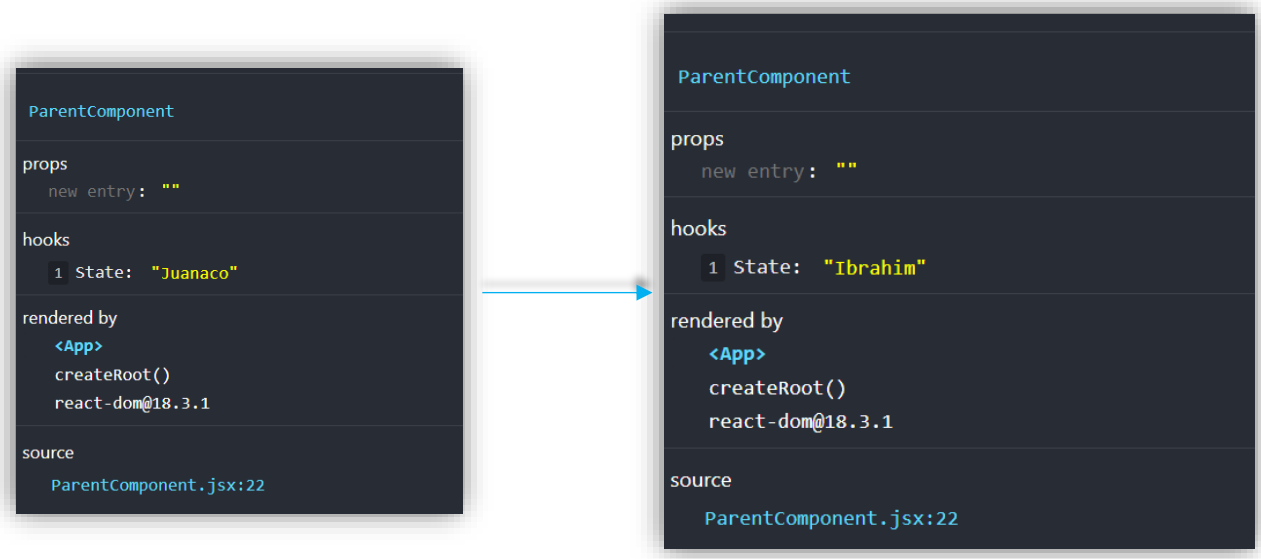


Gracias a esto, he conseguido que el hijo pueda actualizar el estado del padre al hacer clic en un botón. He añadido dos botones: uno que cambia el nombre a "Juanky" y otro que lo cambia a "Ibrahim" para probar que la reactividad funciona perfectamente.

---

4. Inspección con React DevTools (Parte G)

Para terminar, he utilizado la extensión React Developer Tools para inspeccionar mis componentes. He podido navegar por el árbol de jerarquía y observar en tiempo real cómo cambian las props y el estado directamente desde las herramientas de desarrollo.



---

5. Mi Código Fuente (App.jsx)

Este es el código final que he desarrollado integrando todas las partes de la práctica:

```
import ParentComponent from "./ParentComponent"

function App() {
    return <ParentComponent />
}

export default App
```