■ Documentación Técnica — Hooks en React

Introducción

Los Hooks son una característica introducida en React 16.8 que permiten utilizar estado y otras características de React sin escribir clases. Proveen una API más sencilla, expresiva y fácil de reutilizar para manejar la lógica de componentes.

Motivación

Antes de los Hooks, React solo permitía usar estado y ciclo de vida dentro de componentes de clase. Esto provocaba:

- Componentes muy grandes y difíciles de mantener.

- Reutilización limitada de lógica entre componentes.

- Dificultad para organizar código relacionado (efectos, suscripciones, etc.).

Los Hooks resuelven estos problemas al ofrecer una forma funcional y declarativa de trabajar con React.

Reglas de los Hooks

1. ■ Llamar Hooks solo en el nivel superior

- No los uses dentro de bucles, condiciones o funciones anidadas.

- Siempre deben ejecutarse en el mismo orden.

2. ■■ Llamar Hooks solo desde funciones de React

- Pueden usarse en componentes funcionales.

- También en custom Hooks.

Hooks básicos

useState

Permite añadir estado local a un componente funcional.

```
import { useState } from "react";

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

  return (
    <div>
      <p>Has hecho clic {count} veces</p>
      <button onClick={() => setCount(count + 1)}>Incrementar</button>
    </div>
```

```
  );
}
```

useEffect

Maneja efectos secundarios en componentes funcionales (suscripciones, fetch, manipulación del DOM).

```
import { useState, useEffect } from "react";

function Reloj() {
  const [hora, setHora] = useState(new Date());

  useEffect(() => {
    const id = setInterval(() => setHora(new Date()), 1000);
    return () => clearInterval(id); // cleanup
  }, []);

  return <p>{hora.toLocaleTimeString()}</p>;
}
```

useContext

Accede al valor de un contexto sin necesidad de prop drilling.

```
import { useContext } from "react";
import { ThemeContext } from "./ThemeProvider";

function Boton() {
  const theme = useContext(ThemeContext);
  return <button className={theme}>Click</button>;
}
```

Hooks adicionales

- useReducer: Manejo de estado complejo con lógica tipo reducer.
- useCallback: Memoriza funciones para evitar renders innecesarios.
- useMemo: Memoriza valores computados costosos.
- useRef: Acceso a elementos del DOM o valores persistentes.
- useLayoutEffect: Similar a useEffect, pero se ejecuta antes del renderizado.