Hooks Principales en ReactJS

Hook	¿Qué hace?	Importancia técnica
useState	Crea y gestiona estado local en componentes funcionales	🔥 Fundamental para interactividad
useEffect	Ejecuta efectos secundarios (fetch, timers, suscripciones, etc.)	🔥 Clave para lógica reactiva
useContext	Accede a valores globales definidos por React.createContext()	🔄 Útil para evitar prop drilling
useRef	Crea una referencia mutable que persiste entre renders	🧠 Ideal para acceder al DOM o guardar valores
useMemo	Memoriza el resultado de una función costosa	⚡ Optimiza rendimiento
useCallback	Memoriza una función para evitar recrearla en cada render	⚡ Evita renders innecesarios en hijos
useReducer	Alternativa a useState para lógica de estado más compleja	🧩 Útil en simulaciones o lógica tipo Redux
useLayoutEffect	Similar a useEffect, pero se ejecuta **antes** del render visual	🧠 Preciso para mediciones de layout
useImperativeHandle	Expone funciones personalizadas desde componentes con forwardRef	🛠️ Avanzado para control externo de componentes
useDebugValue	Muestra información personalizada en React DevTools	🧪 Útil para debugging de hooks personalizados
useId	Genera un ID único por componente para accesibilidad o claves	🔗 Útil en formularios y listas dinámicas