useRef

> **Prueba la nueva documentación de React para `useRef`.**
> La nueva documentación reemplazará próximamente este sitio, que será archivado. Deja tu opinión aquí

```
const refContainer = useRef(initialValue);
```

`useRef` devuelve un objeto *ref* mutable cuya propiedad `.current` se inicializa con el argumento pasado (`initialValue`). El objeto devuelto se mantendrá persistente durante la vida completa del componente.

Un caso de uso común es para acceder a un hijo imperativamente:

```
function TextInputWithFocusButton() {
  const inputEl = useRef(null);
  const onButtonClick = () => {
    // `current` apunta al elemento de entrada de texto montado
    inputEl.current.focus();
  };
  return (
    <>
      <input ref={inputEl} type="text" />
      <button onClick={onButtonClick}>Focus the input</button>
    </>
  );
}
```

En esencia, `useRef` es como una "caja" que puedes mantener en una variable mutable en su propiedad `.current`.

Puede que estes familiarizado con las referencias principalmente como un medio para acceder al DOM. Si pasas un objeto de referencia a React con `<div ref={myRef} />`, React configurará su propiedad `.current` al nodo del DOM correspondiente cuando sea que el nodo cambie.

Sin embargo, `useRef()` es útil para más que el atributo `ref`. Es conveniente para mantener cualquier valor mutable que es similiar a como usarías campos de instancia en las clases.

Esto funciona debido a que `useRef()` crea un objeto JavaScript plano. La única diferencia entre `useRef()` y crear un objeto `{current: ...}` por ti mismo es que `useRef` te dará el mismo objeto de referencia en cada renderizado.

Ten en cuenta que `useRef` *no* notifica cuando su contenido cambia. Mutar la propiedad `.current` no causa otro renderizado. Si quieres correr algún código cuando React agregue o quite una referencia de un nodo del DOM, puede que quieras utilizar en su lugar una referencia mediante callback.