

1.- crear proyecto:

`npx create-react-app nombreproyecto,`

2.- entrar en el proyecto con:

`cd mover proyecto desde el explorador`

3.- entrar en VSC con:

`core .`

4.- abrir navegador con

`npm start`

5.- crear componentes ¡CON MAYUSCULAS!

5.1 square

5.2 Board

5.3 componente interactivo: Botón

5.4 componente interactivo: Use State (redodar) Set Value

5.5 manejo de estado en el componente Board, el componente padre Board pasa props a los componentes hijos Square para que puedan mostrarse correctamente.

6.-exportar componentes:

`export default`

7.- importar componentes en app.js:

`import`

React DevTools

Para el desarrollo local, React DevTools está disponible como Chrome, Firefox, y Edge extensión del navegador. Después de instalarlo, la pestaña Componentes aparecerá en las Herramientas de desarrollo de tu navegador para los sitios que utilizan React.

El atributo `onClick` del elemento DOM

tiene un significado especial para React porque es un componente integrado. Para componentes personalizados como Square, el nombre depende de ti. Podrías dar cualquier nombre a la prop `onSquareClick` de Square o `handleClick` de Board, y el código funcionaría de la misma manera. En React, es convencional usar nombres `on[Event]` para props que representan eventos y `handle[Event]` para las definiciones de funciones que controlan los eventos.

`handleClick`, llama a `.slice()` para crear una copia de la matriz `squares` en lugar de modificar la matriz existente.

Y así es como se vería si cambiaras los datos sin mutar de la matriz `squares`:

```
const squares = [null, null, null, null, null, null, null, null, null];
const nextSquares = ['X', null, null, null, null, null, null, null, null];
```

// Ahora `squares` no ha cambiado, pero el primer elemento de `nextSquares` es 'X' en lugar de `null`

Importancia de la inmutabilidad 1) la habilidad de deshacer y rehacer ciertas acciones es un requisito común para las aplicaciones. 2) La inmutabilidad hace que sea muy barato para los componentes comparar si sus datos han cambiado o no. Puedes obtener más información sobre cómo React elige cuándo volver a renderizar un componente en la documentación de referencia de la API `memo`.