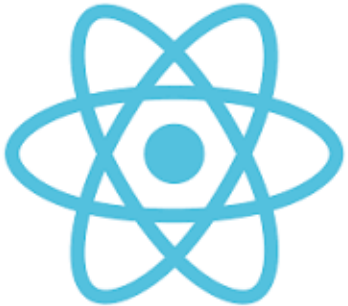
<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

ESPECIALIZACIÓN REACT



BA

<codoa codo/>

Clase 09. REACT JS

EVENTOS

BA

Responder a eventos

React te permite añadir controladores de eventos a tu JSX. Los controladores de eventos son tus propias funciones que se ejecutarán en respuesta a interacciones como hacer clic, hover, enfocar inputs en formularios, entre otras.

Añadiendo controladores de eventos
Para agregar un controlador de evento, primero definirás una función y luego la pasarás como una prop a la etiqueta JSX apropiada. Por ejemplo, este es un botón que no hace nada todavía:

```
export default function Button() {
  return (
    <button>
      No hago nada
    </button>
  );
}
```

Puedes hacer que muestre un mensaje cuando un usuario haga clic siguiendo estos tres pasos:

Declara una función llamada handleClick dentro de tu componente Button.
Implementa la lógica dentro de esa función (utiliza alert para mostrar el mensaje).
Agrega onClick={handleClick} al JSX del <button>.

```
export default function Button() {
  function handleClick() {
    alert('¡Me hiciste clic!');
  }

  return (
    <button onClick={handleClick}>
      Hazme clic
    </button>
  );
}
```

Definiste la función handleClick y luego la pasaste como una prop al <button>. handleClick es un controlador de evento. Las funciones controladoras de evento:

Usualmente están definidas dentro de tus componentes.
Tienen nombres que empiezan con handle, seguido del nombre del evento.
Por convención, es común llamar a los controladores de eventos como handle seguido del nombre del evento. A menudo verás onClick={handleClick}, onMouseEnter={handleMouseEnter}, etcétera.

Por otro lado, puedes definir un controlador de evento en línea en el JSX:

```
<button onClick={function handleClick() {
  alert('¡Me hiciste clic!');
}}>
```

O, de manera más corta, usando una función flecha:

```
<button onClick={() => {
  alert('¡Me hiciste clic!');
}}>
```

ATENCIÓN

Las funciones que se pasan a los controladores de eventos deben ser pasadas, no llamadas. Por ejemplo:

pasar una función (correcto)
```
<button onClick={handleClick}>
```

llamar una función (incorrecto)
```
<button onClick={handleClick()}>
```

La diferencia es sutil. En el primer ejemplo, la función handleClick es pasada como un controlador de evento onClick. Esto le dice a React que lo recuerde y solo llama la función cuando el usuario hace clic en el botón.

En el segundo ejemplo, los () al final del handleClick() ejecutan la función inmediatamente mientras se renderiza, sin ningún clic. Esto es porque el JavaScript dentro de { y } en JSX se ejecuta de inmediato.

Cuando escribes código en línea, la misma trampa se presenta de otra manera:

pasar una función (correcto)
<button onClick={() => alert('...')}>

llamar una función (incorrecto)
<button onClick={alert('...')}>

Pasar código en línea así no lo ejecutará al hacer clic; lo ejecutará cada vez que el componente se renderice:

// Esta alerta se ejecuta cuando el componente se renderiza, no cuando ¡hiciste clic!
<button onClick={alert('¡Me hiciste clic!')}>
Si quieres definir un controlador de evento en línea, envuélvelo en una función anónima de esta forma:

<button onClick={() => alert('¡Me hiciste clic!')}>
En lugar de ejecutar el código que está dentro cada vez que se renderiza, esto crea una función para que se llame más tarde.

En ambos casos, lo que quieres pasar es una función:

<button onClick={handleClick}> pasa la función handleClick.
<button onClick={() => alert('...')}> pasa la función () => alert('...').