

¿Qué son los Event Handlers?

- Son funciones que se activarán tras la interacción del usuario con un elemento del DOM: click, hover, form submit, input change, etc.
- En React, los componentes cuentan con un atributo para escuchar un evento de interacción del usuario como alternativa a usar addEventListener()

```
// Functional components
<form onSubmit={handleSubmit}>
    <button type="submit">Submit</button>
</form>
```

El atributo que escucha el evento recibe una función que maneja este evento y ejecuta el código que deseamos ejecutar tras la acción

```
const handleSubmit = (e) => {
  e.preventDefault();
  // TODO: handle submit data
}
```

https://react.dev/learn/adding-interactivity



• En el atributo de evento podemos pasarle como argumento una función que ejecute otra

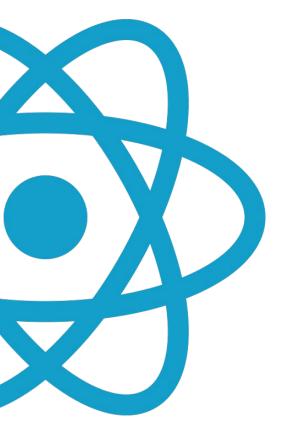
```
<button onClick={(e) => handleEditRow(row, e)}>Edit Row</button>
```

• En el atributo de evento NO debemos ejecutar una función directamente

```
// correcto
<button onClick={handleClick}>Click</button>
// incorrecto
<button onClick={handleClick()}>
```

https://react.dev/learn/adding-interactivity





¿Qué son los hooks en React?

- Son funciones nativas o personalizadas de react que permiten usar las características de los componentes funcionales
- Los hooks nativos más usados de React son:
 - useState()
 - useEffect()
 - useRef()
 - useld()
 - useCallback()
 - useContext()
 - useReducer()
 - useMemo()

https://react.dev/reference/react