



# Arrays

clase n° 13

\*REC



recordá  
poner a grabar la clase

# eventos en JSX

Vamos a arrancar hablando de los eventos. ¿Qué son? Básicamente, son las interacciones que los usuarios tienen con nuestra página: un clic, escribir en un formulario, mover el mouse, son lo que hace que nuestras aplicaciones sean interactivas, algo vital hoy en día.


Ahora, si ya usaste eventos en HTML, como ese famoso **onclick** en minúsculas, en JSX es un poquito diferente. Acá usamos camelCase: **onClick**, **onChange**, etc. Además, no vamos a escribir cosas como **onclick="miFuncion()"**, sino que directamente pasamos una función dentro del atributo. Te prometo que es más fácil de mantener.

## Diferencias:



```
1 <button onclick="alert('Hola!')">Hace clic</button>
```

HTML



```
1 <button onClick={() => alert('Hola!')}>Hace clic</button>
```

JSX

# eventos en JSX en profundidad

**onClick:** Cuando alguien hace clic en algo, React se entera. Y para que "reaccione", le decimos qué hacer con `onClick`. Por ejemplo, imaginemos un contador:

```
1 import React, { useState } from "react";
2 function Contador() {
3   const [contador, setContador] = useState(0);
4   const incrementar = () => setContador(contador + 1);
5   return (
6     <div>
7       <p>Contador: {contador}</p>
8       <button onClick={incrementar}>Incrementar</button>
9     </div>
10  );
11 }
12 export default Contador;
```

# eventos en JSX en profundidad

**onChange:** Acá es donde entran los formularios. Si queremos captar lo que alguien escribe en un input, usamos onChange:

```
1 import React, { useState } from "react";
2 function InputDinamico() {
3   const [texto, setTexto] = useState("");
4   const manejarCambio = (e) => setTexto(e.target.value);
5   return (
6     <div>
7       <input type="text" onChange={manejarCambio} placeholder="Escribí algo" />
8       <p>Texto: {texto}</p>
9     </div>
10  );
11 }
12 export default InputDinamico;
```

# eventos en JSX en profundidad

**event.target:** Para acceder al elemento que disparó el evento, usamos event.target. Por ejemplo, cambiemos el color de fondo de un div dependiendo del color que el usuario elija:

```
1 import React, { useState } from "react";
2 function CambiarColor() {
3   const [color, setColor] = useState("#ffffff");
4   const manejarCambio = (e) => setColor(e.target.value);
5   return (
6     <div style={{ backgroundColor: color, height: "100vh" }}>
7       <input type="color" onChange={manejarCambio} />
8       <p>El color actual es: {color}</p>
9     </div>
10  );
11 }
12 export default CambiarColor;
```

**Ampliación completa:** Vamos a hacer un formulario interactivo. La idea es capturar un nombre y un email, y cuando presionen un botón, mostrar un mensaje agradeciéndoles.

```
1 import React, { useState } from "react";
2 function FormularioInteractivo() {
3   const [nombre, setNombre] = useState("");
4   const [email, setEmail] = useState("");
5   const [mensaje, setMensaje] = useState("");
6   const enviarFormulario = () => {
7     setMensaje(`Gracias ${nombre}! Te contactaremos en ${email}.`);
8   };
9   return (
10     <div>
11       <input
12         type="text"
13         placeholder="Nombre"
14         value={nombre}
15         onChange={(e) => setNombre(e.target.value)}
16       />
17       <input
18         type="email"
19         placeholder="Email"
20         value={email}
21         onChange={(e) => setEmail(e.target.value)}
22       />
23       <button onClick={enviarFormulario}>Enviar</button>
24       <p>{mensaje}</p>
25     </div>
26   );
27 }
28 export default FormularioInteractivo;
```



revolución\*  
digital\_