

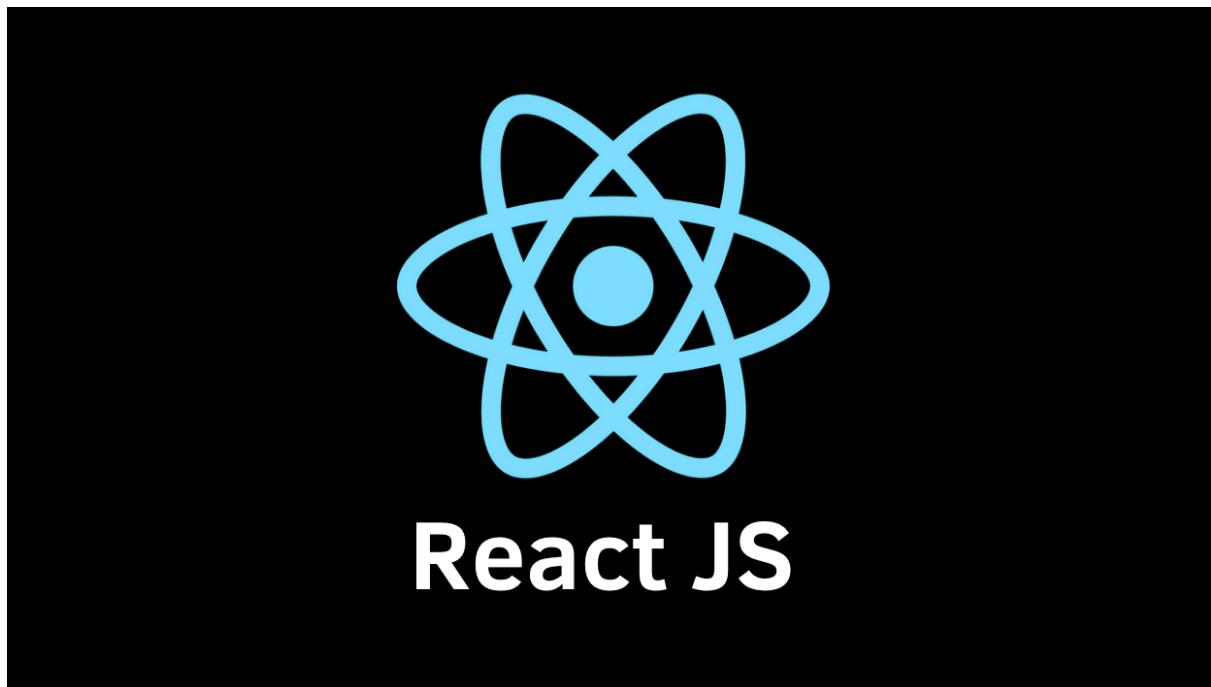
Desarrollo web en entorno cliente -

Nivel 6 React

Pablo Valero López

22/01/2026

2ºDAW



Índice:

Objetivo

[Parte A: onClick y el objeto event](#)

[Parte B: Cambiar la UI con estado + evento](#)

[Parte C: onChange \(entrada controlada\)](#)

[Parte D: onSubmit \(formulario\)](#)

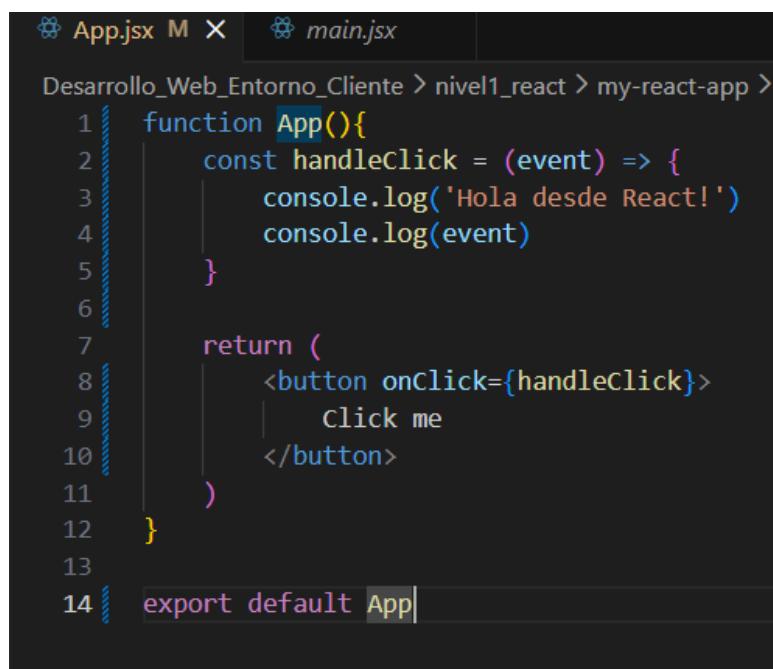
[Miniretillo \(pero esta vez sin ayuda\)](#)

Objetivo

En esta práctica vas a aprender a reaccionar a interacciones del usuario (click, cambio de texto y envío de formularios) usando manejadores de eventos en React. Verás cómo React te entrega un evento sintético y cómo combinar eventos con estado para actualizar la interfaz de manera dinámica.

Parte A: onClick y el objeto event

En la siguiente imagen se muestra el código de onClick.



The screenshot shows a code editor with two tabs: 'App.jsx' and 'main.jsx'. The 'App.jsx' tab is active and displays the following code:

```
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app >
1  function App(){
2      const handleClick = (event) => {
3          console.log('Hola desde React!')
4          console.log(event)
5      }
6
7      return (
8          <button onClick={handleClick}>
9              Click me
10             </button>
11         )
12     }
13
14 export default App|
```

Imagen 1: Código de onClick. Elaboración propia.

En las siguientes imágenes se muestran los resultados de onClick.

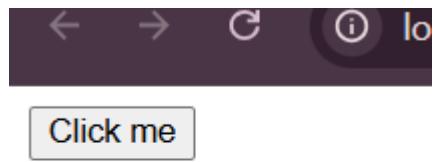


Imagen 2: Resultado de onClick. Elaboración propia.

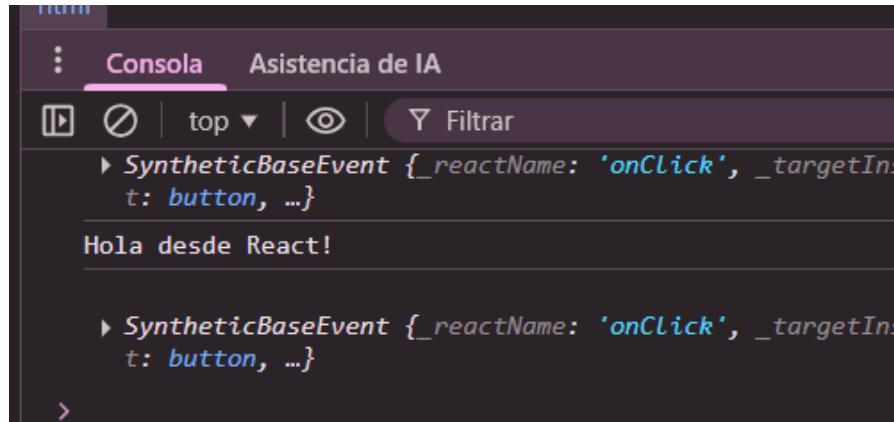


Imagen 3: Resultado de onClick. Elaboración propia.

Parte B: Cambiar la UI con estado + evento

En la siguiente imagen se muestra el código para cambiar el estado.

```
App.jsx M main.jsx
Desarrollo_Web_Entorno.Cliente > nivel1_react > my-react-app > src > App.jsx > App
1 import {useState} from 'react'
2
3 function App(){
4   const [isParagraphVisible, setIsParagraphVisible] = useState(true)
5
6   const toggleStatus = () => {
7     setIsParagraphVisible(!isParagraphVisible)
8   }
9
10  return (
11    <>
12      <h1>Change UI based on click</h1>
13      {isParagraphVisible && (
14        <p>This paragraph will be shown/hidden on click</p>
15      )}
16
17      <button onClick={toggleStatus}>
18        {isParagraphVisible ? 'Hide' : 'Show'} Paragraph
19      </button>
20    </>
21  )
22}
23
24 export default App
```

Imagen 4: Código de evento. Elaboración propia.

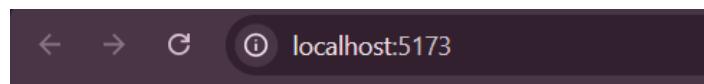
En la siguientes imágenes se muestran el resultado del evento.



Change UI badges on click

This paragraph will be shown/hidden on click

Imagen 5: Código de evento. Elaboración propia.



Change UI badges on click

Imagen 6: Código de evento. Elaboración propia.

Parte C: onChange (entrada controlada)

En la siguiente imagen se muestra el código de una entrada controlada.

```
App.jsx M X main.jsx
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > default
1 import {useState} from 'react'
2
3 function App(){
4     const [nickname, setNickname] = useState('')
5
6     const toggleStatus = () => {
7         setIsParagraphVisible(!isParagraphVisible)
8     }
9
10    return (
11        <>
12            <h1>Alias del jugador</h1>
13            <input type="text"
14                placeholder='Escribe tu alias'
15                value={nickname}
16                onChange={(e)>setNickname(e.target.value)} />
17            <p>Tu alias es: <strong>{nickname || '...'}</strong></p>
18        </>
19    )
20
21
22 export default App
```

Imagen 6: Código de entrada controlada. Elaboración propia.

En las siguientes imágenes se muestran los resultados de entrada controlada



Alias del jugador

Tu alias es: ...

Imagen 6: Resultado de entrada controlada. Elaboración propia.



Alias del jugador

Tu alias es: Klara

Imagen 7: Resultado de entrada controlada. Elaboración propia.

Parte D: onSubmit (formulario)

En la siguiente imagen se muestra el código con el formulario.

```
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > [1] import {useState} from 'react'  
[2]  
[3] function App(){  
[4]     const [email, setEmail] = useState('')  
[5]     const [msg, setMsg] = useState('')  
[6]  
[7]     const handleSubmit = (e) => {  
[8]         e.preventDefault()  
[9]         setMsg(`Inscripción enviada para: ${email}`)  
[10]    }  
[11]  
[12]    return (  
[13]        <>  
[14]        <h1>Registro</h1>  
[15]        <form onSubmit={handleSubmit}>  
[16]            <input  
[17]                type="email"  
[18]                placeholder='tu@mail.com'  
[19]                value={email}  
[20]                onChange={(e)=>setEmail(e.target.value)}  
[21]                required  
[22]            />  
[23]            <button type='submit'>Enviar</button>  
[24]        </form>  
[25]  
[26]        {msg && <p>{msg}</p>}  
[27]    </>  
[28]    )  
[29] }  
[30]  
[31] export default App
```

Imagen 8: Código de formulario. Elaboración propia.

En la siguiente imagen se muestra el resultado del formulario.

The screenshot shows a registration form titled "Registro". It has a single input field containing the email "klara@mail.com" and a blue "Enviar" button. Below the form, a message says "Inscripción enviada para: klara@mail.com".

Imagen 9: Resultado de formulario. Elaboración propia.

Miniretillo (pero esta vez sin ayuda)

En las siguientes imágenes se muestra el código de minireto

The screenshot shows a code editor with the file "App.jsx" open. The code uses the useState hook from React to manage state for three variables: "isParagraphVisible", "newContador", and "estadoBoton".

```
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > App
1 import {useState} from 'react'
2
3 function App(){
4     const [isParagraphVisible, setIsParagraphVisible] = useState(true)
5     const toggleStatus = () => {
6         setIsParagraphVisible(!isParagraphVisible)
7     }
8
9     const [newContador, setNewContador] = useState(0)
10    const [modo, setModo] = useState()
11
12    const [estadoBoton, setEstadoBoton] = useState(true)
13    const cambiarBoton = () => {
14        setEstadoBoton(!estadoBoton)
15    }
16}
```

Imagen 10: Código minireto. Elaboración propia.

```

        return (
            <>
            <h1>Miniretillo</h1>
            {isParagraphVisible && (
                <p>Este es un texto oculto.</p>
            )}
            <button onClick={toggleStatus}>
                {isParagraphVisible ? 'Esconder' : 'Mostrar'} texto
            </button>
            <p>Contador {newContador}</p>
            <button onClick={() => setNewContador((n) => n + 1)}>+1 numero</button>
            <button onClick={() => setNewContador(0)}>Reset</button>
            <br/>
            <br/>
            <input type="text" value={modo} placeholder='Añade algo (papafrita)' onChange={(e)=>setModo(e.target.value)}/>
            {modo && <p>{modo}</p>}
            <br/>
            <br/>
            {estadoBoton && (
                <p>Este botón ya estaba al inicio.</p>
            )}
            <button onClick={cambiarBoton}>{estadoBoton ? 'Ocultar' : 'Mostrar'}</button>
        )
    )
}

export default App

```

Imagen 11: Código miniretro. Elaboración propia.

En la siguiente imagen se muestra el resultado del miniretro.



Miniretillo

Este es un texto oculto.

Esconder texto

Contador 3

+1 numero Reset

Papafrita

Papafrita

Este botón ya estaba al inicio.

Ocultar

Imagen 12: Resultado miniretro. Elaboración propia.