
TAREAS REACT - NIVEL 6

Tabla de contenido

Parte A: onClick y el objeto evento.....	2
Parte B: Cambiar la UI con estado + evento	2
Parte C: onChange (entrada controlada).....	3
.....	3
.....	3
Parte D: onSubmit (formulario)	3
Miniretillo:	4
Preguntas cortas:	4

Parte A: onClick y el objeto evento

En la siguiente imagen se puede ver como con cada pulsación se genera el mensaje con el código de la ficha:

The screenshot shows the VS Code interface with the following details:

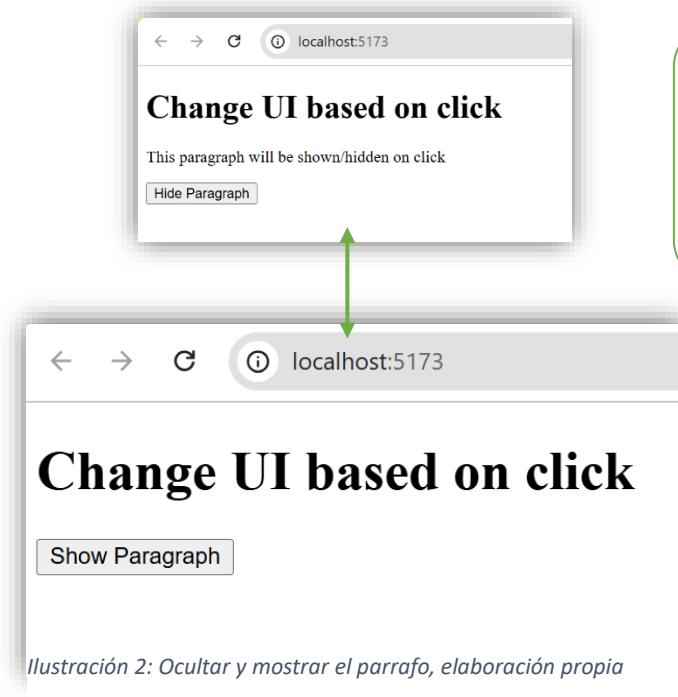
- File Explorer:** Shows files: main.jsx, App.jsx (selected), ChildComponent.jsx, and App.css.
- Code Editor:** Displays the following code from App.jsx:

```
src > App.jsx > ...
You, 13 seconds ago | 1 author (You)
1 function App() {
2   const handleClick = (event) => {
3     console.log('Hola desde React!') ×2 'Hola desde React!' ⏱ 'Hola desde React!'
4     console.log(event) ×2 SyntheticBaseEvent { _reactName: 'onClick', _targetInst: null, ty...
5   }
}
```
- Console:** Shows the following log entries:

Time	Message	Location
17:28:47.955	'Hola desde React!'	src/app.jsx:3
17:28:47.955	SyntheticBaseEvent { _reactName: 'onClick', _targetInst: null, type: 'click', nativeEvent: PointerEvent { pointerId: 1, width: 1, height: 1, pressure: 0, tiltX: 0, tiltY: 0, azimuthAngle: 0, altitudeAngle: 1.5707963267948966, tangentialPressure: 0, } }	src/app.jsx:4
17:29:05.526	'Hola desde React!'	src/app.jsx:3
17:29:05.528	SyntheticBaseEvent { _reactName: 'onClick', _targetInst: null, type: 'click', nativeEvent: PointerEvent { pointerId: 1, width: 1, height: 1, pressure: 0, tiltX: 0, tiltY: 0, azimuthAngle: 0, altitudeAngle: 1.5707963267948966, tangentialPressure: 0, } }	src/app.jsx:4

Ilustración 1: Resultado en la consola, elaboración propia

Parte B: Cambiar la UI con estado + evento



Se gestiona la visibilidad del párrafo mediante "isParagraphVisible", que se invierte al pulsar el botón y utiliza el operador && para mostrar el texto.

Ilustración 2: Ocultar y mostrar el parrafo, elaboración propia

Parte C: onChange (entrada controlada)

A screenshot of a web browser window titled "localhost:5173". The main content area displays the heading "Alias del jugador" in large bold letters. Below it, there is an input field containing the text "Ibra" and a line of text stating "Tu alias es: Ibra". A green callout bubble points from the right side of the browser window towards the explanatory text on the right.

En este caso usamos el evento “onChange” para cuando haya un cambio se actualice el output.

Ilustración 3: Prueba del input, elaboración propia

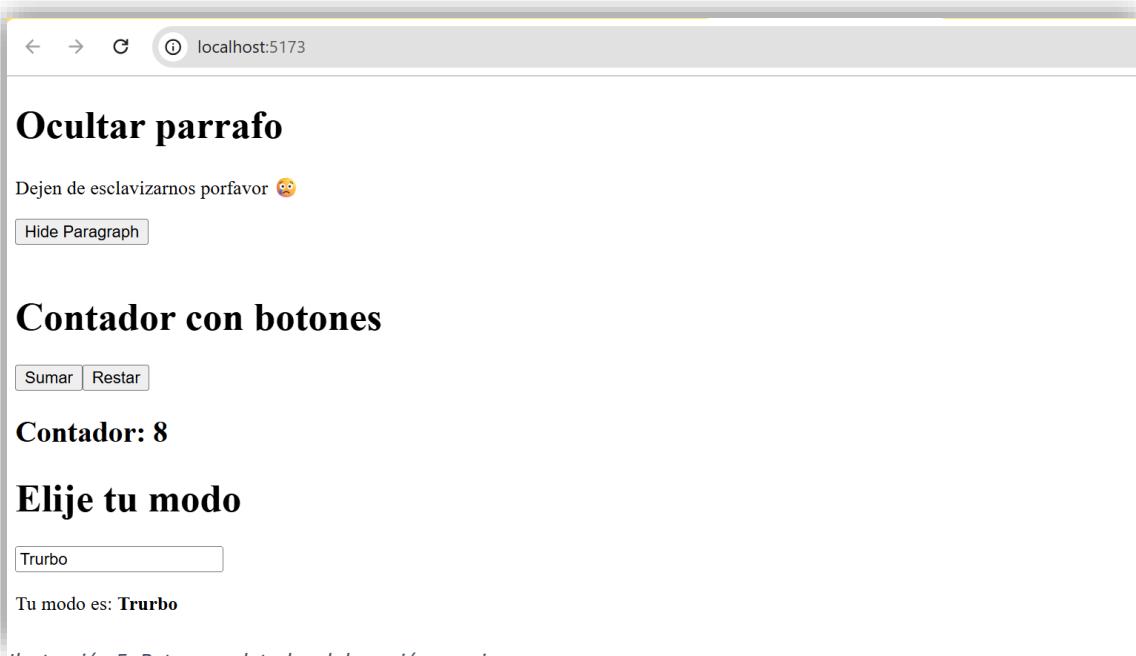
Parte D: onSubmit (formulario)

A diferencia del apartado anterior, en este caso usamos un botón para mandar y mostrar el input.

A screenshot of a web browser window titled "localhost:5173". The main content area displays the heading "Registro" in large bold letters. Below it, there is an input field containing the email address "ibrahimhanaoui1@gmail.co" and a button labeled "Enviar". A line of text below the form states "Inscripción enviada para: ibrahimhanaoui1@gmail.com". A green callout bubble points from the left side of the browser window towards the explanatory text on the left.

Ilustración 4: Prueba del input, elaboración propia

Miniretillo:



Para este reto, he reciclado código visto en otros ejercicios y adaptándolo a lo que me exigía el reto, e disfrutado y entendido de manera más detallada que hace cada elemento para poder adaptarlo, ya que realmente no he tenido que crear código nuevo.

Preguntas cortas:

1. **¿Qué diferencia hay entre onClick y onSubmit?** El **onClick** se ejecuta cuando haces click pero en cambio el **onSubmit** se va ejecutando conforme hayan cambios.
2. **¿Por qué usamos e.preventDefault() en un formulario?** Para evitar que la página se recargue al enviar el formulario y permitir que React gestione los datos.
3. **¿Qué es una "entrada controlada"?** Es un input cuyo valor depende del estado **value** y se actualiza mediante un evento **onChange**, dando a React el control total de lo que se escribe.
4. **¿Qué estados y eventos manejas?**

En el mini-reto manejo tres estados: `isParagraphVisible` , `newContador` y `modo`. Estos se actualizan respectivamente mediante eventos **onClick** en los botones de visibilidad y del contador, y a través del evento **onChange** en el campo de entrada de texto.