

ÍNDICE

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

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

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

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

[Preguntas:](#)

Parte A: onClick y el objeto event:

The screenshot shows a browser window with the URL `localhost:5173`. On the left, the developer tools' console tab displays the following logs:

```
getIdsFromUrl result - promptId: null jobID: null
agentMode: false
Checking domain: localhost
Delay detection: URL parameter 'null', using 1000ms
delay
Queue initialized with delay feature active (1000ms between prompts)
IMPORTANT! Make sure URL has delay parameter if you want to customize delay time
Queue initialization complete - ready to process
messages
Hola desde React!
SyntheticBaseEvent
Hola desde React!
SyntheticBaseEvent
```

On the right, a code editor shows the `App.jsx` file:

```
function App() {
  const handleClick = (event) => {
    console.log('Hola desde React!')
    console.log(event)
  }

  return (
    <button onClick={handleClick}>
      Click me
    </button>
  )
}

export default App
```

(Captura del código por elaboración propia)

Parte B: Cambiar la UI con estado + evento:

The screenshot shows a browser window with the URL `localhost:5173`. On the left, the developer tools' console tab displays the following logs:

```
This paragraph will be shown/hidden on click
Hide Paragraph
```

On the right, a code editor shows the `App.jsx` file:

```
import { useState } from 'react'

function App() {
  const [isParagraphVisible, setIsParagraphVisible] = useState(true)
  const toggleStatus = () => {
    setIsParagraphVisible(!isParagraphVisible)
  }

  return (
    <h1>Change UI based on click</h1>
    <div>
      {isParagraphVisible && (
        <p>This paragraph will be shown/hidden on click</p>
      )}
      <button onClick={toggleStatus}>
        {isParagraphVisible ? 'Hide' : 'Show'} Paragraph
      </button>
    </div>
  )
}

export default App
```

Below this, another code editor window shows the `main.jsx` file:

```
import { useState } from 'react'

function App() {
  const [isParagraphVisible, setIsParagraphVisible] = useState(true)
  const toggleStatus = () => {
    setIsParagraphVisible(!isParagraphVisible)
  }

  return (
    <h1>Change UI based on click</h1>
    <div>
      {isParagraphVisible && (
        <p>This paragraph will be shown/hidden on click</p>
      )}
      <button onClick={toggleStatus}>
        {isParagraphVisible ? 'Hide' : 'Show'} Paragraph
      </button>
    </div>
  )
}

export default App
```

(Captura del código por elaboración propia)

Parte C: onChange (entrada controlada):

The screenshot shows a browser window with the URL `localhost:5173`. On the left, the developer tools' console tab displays the following logs:

```
Hola Klara
'u alias es: Hola Klara
```

On the right, a code editor shows the `App.jsx` file:

```
import { useState } from 'react'

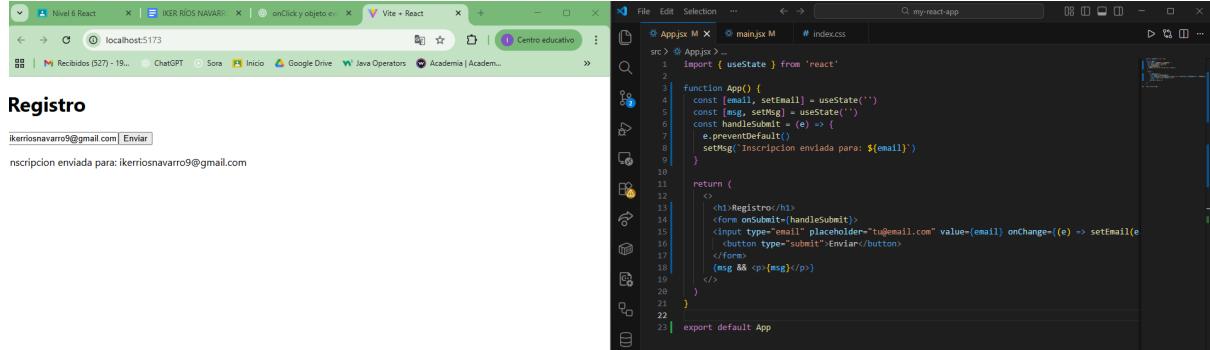
function App() {
  const [nickname, setNickname] = useState('')
  const handleChange = (e) => {
    setNickname(e.target.value)
  }

  return (
    <h1>Alias del jugador</h1>
    <input type="text" placeholder="Escribe tu alias" value={nickname} onChange={handleChange} />
    <p>Tu alias es: <strong>{nickname || '...'}</strong></p>
  )
}

export default App
```

(Captura del código por elaboración propia)

Parte D: onSubmit (formulario):



The screenshot shows a browser window with a registration form titled "Registro". The form has an input field with the value "ikerriosnavarro9@gmail.com" and a button labeled "Enviar". Below the form, a message says "Inscripción enviada para: ikerriosnavarro9@gmail.com". To the right of the browser is a code editor displaying a React component named "App.js". The code uses the useState hook to manage state for email and message. It includes a handleSubmitted function that logs the email to the console and sends an email using the Mail API. The onSubmit event is handled by the handleSubmitted function.

```
src/App.js
import { useState } from 'react'

function App() {
  const [email, setEmail] = useState('')
  const [msg, setMsg] = useState('')
  const handleSubmitted = (e) => {
    e.preventDefault()
    console.log(`Inscripción enviada para: ${email}`)
  }

  return (
    <h1>Registro</h1>
    <form onSubmit={handleSubmitted}>
      <input type="email" placeholder="tu@email.com" value={email} onChange={(e) => setEmail(e.target.value)} />
      <button type="submit">Enviar</button>
    </form>
    {msg && <p>{msg}</p>}
  )
}

export default App
```

(Captura del código por elaboración propia)

Preguntas:

1. ¿Qué diferencia hay entre onClick y onSubmit?

onClick actúa sobre elementos individuales, mientras que onSubmit gestiona el envío completo de un formulario.

2. ¿Por qué usamos e.preventDefault() en un formulario?

preventDefault() evita el comportamiento por defecto del navegador y permite controlar el formulario desde React.

3. ¿Qué es una "entrada controlada" y por qué usamos value + onChange?

Usamos value + onChange para que React tenga el control total del input.

4. En tu mini-reto, que estado(s) manejas y que evento(s) los actualizan?

Los estados se actualizan mediante eventos onClick asociados a botones que modifican el estado con setState.