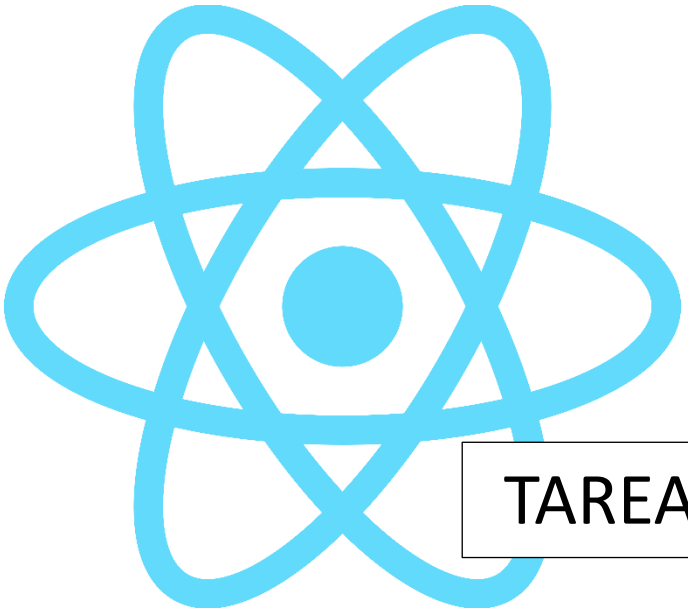

TAREA 6: REACT

Contenido

Parte A: onClick y el objeto event

```jsx
function App() {
  const handleClick = (event) => {
    console.log('Hola desde React!')
    console.log(event)
  }
  return (
    <button onClick={handleClick}>
    Click me
    </button>
  )
}
export default App
```

Ilustración 1. onClick y el event. Elaboración propia

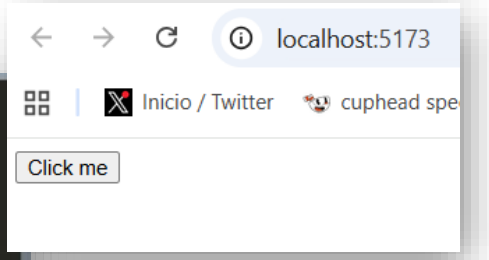

Ilustración 2. Visualización del resultado. Elaboración propia

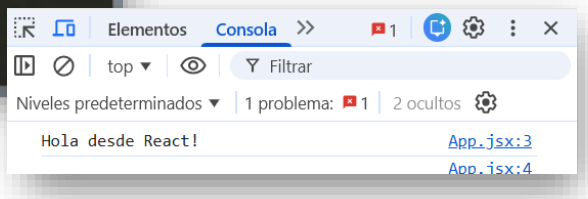

Ilustración 3. Visualización del resultado en la consola. Elaboración propia

Parte B: Cambiar la UI con estado + evento

```jsx
import { useState } from 'react'
function App() {
  const [isParagraphVisible, setIsParagraphVisible] = useState(true)
  const toggleStatus = () => {
    setIsParagraphVisible(!isParagraphVisible)
  }
  return (
    <>
      <h1>Change UI based on click</h1>
      {isParagraphVisible && (
        <p>This paragraph will be shown/hidden on click</p>
      )}
      <button onClick={toggleStatus}>
        {isParagraphVisible ? 'Hide' : 'Show'} Paragraph
      </button>
    </>
  )
}
export default App
```

Ilustración 4. Cambiar la UI. Elaboración propia

Ilustración 5. Visualización del resultado. Elaboración propia

Parte C: onChange (entrada controlada)

```jsx
import { useState } from 'react'
function App() {
  const [nickname, setNickname] = useState('')

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

Ilustración 6. onChange. Elaboración propia

Ilustración 7. Visualización del resultado. Elaboración propia

Parte D: onSubmit (formulario)

```jsx
import { useState } from 'react'
function App() {
  const [email, setEmail] = useState('')
  const [msg, setMsg] = useState('')
  const handleSubmit = (e) => {
    e.preventDefault()
    setMsg(`Inscripcion enviada para: ${email}`)
  }
  return (
    <>
      <h1>Registro</h1>
      <form onSubmit={handleSubmit}>
        <input
          type="email"
          placeholder="tu@email.com"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
        <button type="submit">Enviar</button>
      </form>
      {msg && <p>{msg}</p>}
    </>
  )
}
export default App
```

Ilustración 8. onSubmit. Elaboración propia

Ilustración 9. Visualización del resultado. Elaboración propia

Miniretillo (pero esta vez sin ayuda)

App.jsx

```jsx
import Panel from './Panel'
import Contador from './Contador'

function App() {
  return (
    <div style={{ padding: '20px' }}>
      <h1>Mi Panel de Control</h1>

      {/* Llamamos al componente Panel (Input + Mostrar/Ocultar) */}
      <Panel />

      {/* Línea divisoria para separar las secciones */}
      <hr />

      {/* Llamamos al componente Contador (+1 / -1) */}
      <Contador />
    </div>
  )
}

export default App
```

Ilustración 10. App.jsx. Elaboración propia

Ilustración 11. Visualización del resultado. Elaboración propia

Contador.jsx

```jsx
import { useState } from 'react'

function Contador() {
    // Definimos el estado 'count' empezando en 0
    const [count, setCount] = useState(0)

    // Función para aumentar el contador
    const sumar = () => setCount(count + 1)

    // Función para restar con la condición de no bajar de 0
    const restar = () => {
        if (count > 0) {
            setCount(count - 1)
        }
    }

    return (
        <div style={{ marginTop: '20px' }}>
            {/* Mostramos el valor del estado en la interfaz */}
            <h3>Contador: {count}</h3>

            {/* onClick para ejecutar las funciones de arriba */}
            <button onClick={sumar}>+1</button>
            <button onClick={restar}>-1</button>
        </div>
    )
}

export default Contador
```

Ilustración 12. Contador.jsx. Elaboración propia

Panel.jsx

```jsx
import { useState } from 'react'

function Panel() {
    // Estado para guardar el texto del input
    const [modo, setModo] = useState('chill')

    // Estado booleano para mostrar u ocultar
    const [mostrar, setMostrar] = useState(true)

    return (
    <div>
        {/* Input controlado: onChange actualiza el estado 'modo' al escribir */}
        <input
        type="text"
        value={modo}
        onChange={(e) => setModo(e.target.value)}
        />
        {/* Mostramos el texto del estado en tiempo real */}
        <p>Modo: <strong>{modo}</strong></p>

        {/* Botón con toggle: cambia el estado 'mostrar' al contrario de lo que tenga */}
        <button onClick={() => setMostrar(!mostrar)}>
        {/* El texto del botón cambia según el estado (Punto opcional) */}
        {mostrar ? 'Ocultar' : 'Mostrar'} sección
        </button>

        {/* Renderizado condicional: si 'mostrar' es true, enseña el párrafo */}
        {mostrar && <p>Esta es la sección que se oculta.</p>}
    </div>
    )
}

export default Panel
```

Ilustración 13. Panel.jsx. Elaboración propia

Preguntas

1. ¿Qué diferencia hay entre onClick y onSubmit?

onClick se activa al pulsar cualquier elemento (como un botón), mientras que onSubmit se activa específicamente al enviar un formulario (al pulsar "Enter" o un botón de tipo submit).

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

Se usa para evitar que la página se recargue automáticamente al enviar el formulario, permitiendo que React maneje los datos sin perder el estado actual.

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

Es un input cuyo valor lo manda React a través del estado. Usamos value para mostrar el estado actual y onChange para actualizar dicho estado cada vez que el usuario escribe.

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

Manejo tres estados: count (número), modo (texto) y mostrar (booleano). Los actualizo con eventos onClick en los botones y onChange en el input de texto.