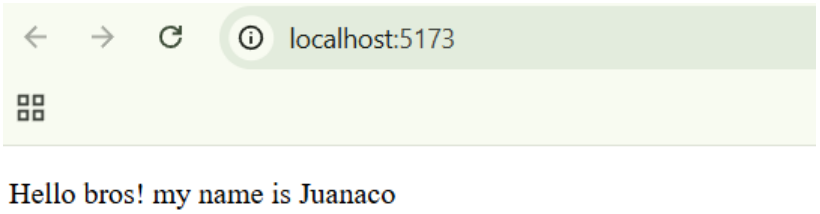
Juan Antonio Zamora Nieto | 2º DAW | 16/01/2026

REACT NIVEL 4

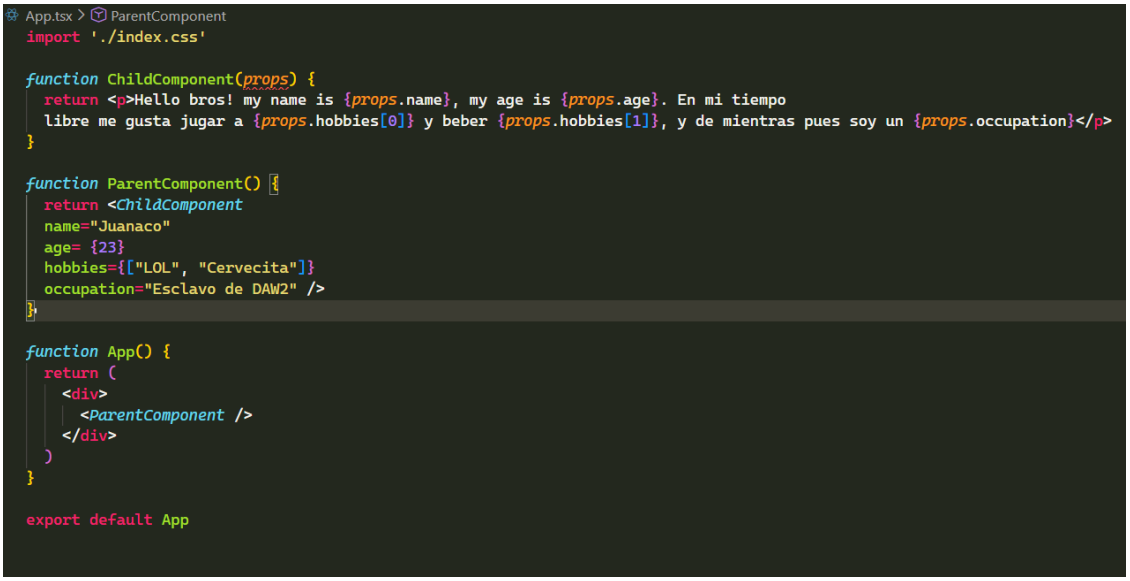
PARTE A: Props básicas (Padre -> Hijo)

(Ilustración 1. Captura de pantalla del código creando dos componentes y pasando una prop llamada name desde el padre, el hijo recibe ese prop y lo muestra por pantalla. Elaboración propia.)

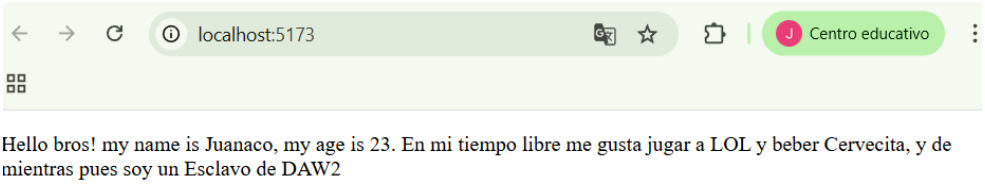


(Ilustración 2. Captura de pantalla de la aplicación mostrando el resultado del primer ejercicio tras crear dos componentes padre e hijo y pasar props del padre al hijo. Elaboración propia.)

PARTE B: Pasar múltiples props



(Ilustración 3. Captura de pantalla del código, en este caso, pasando varios props y mostrando esos varios props que le pasa el padre al hijo en nuestro proyecto. Elaboración propia.)



(Ilustración 4. Captura de pantalla de la aplicación mostrando la frase con los props heredados del `ParentComponent`. Elaboración propia.)

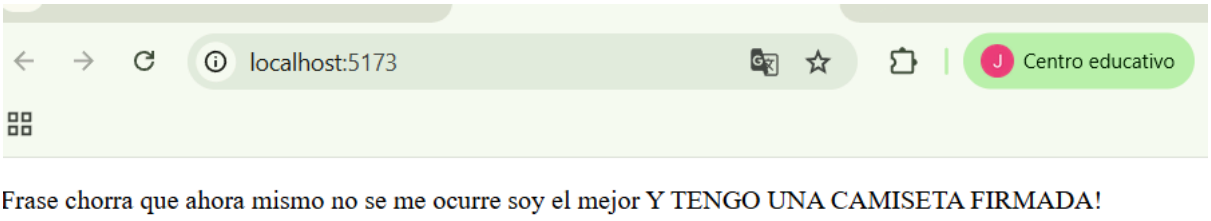
PARTE C: Pasar funciones como props

index.css | App.tsx 1

src > App.tsx > App

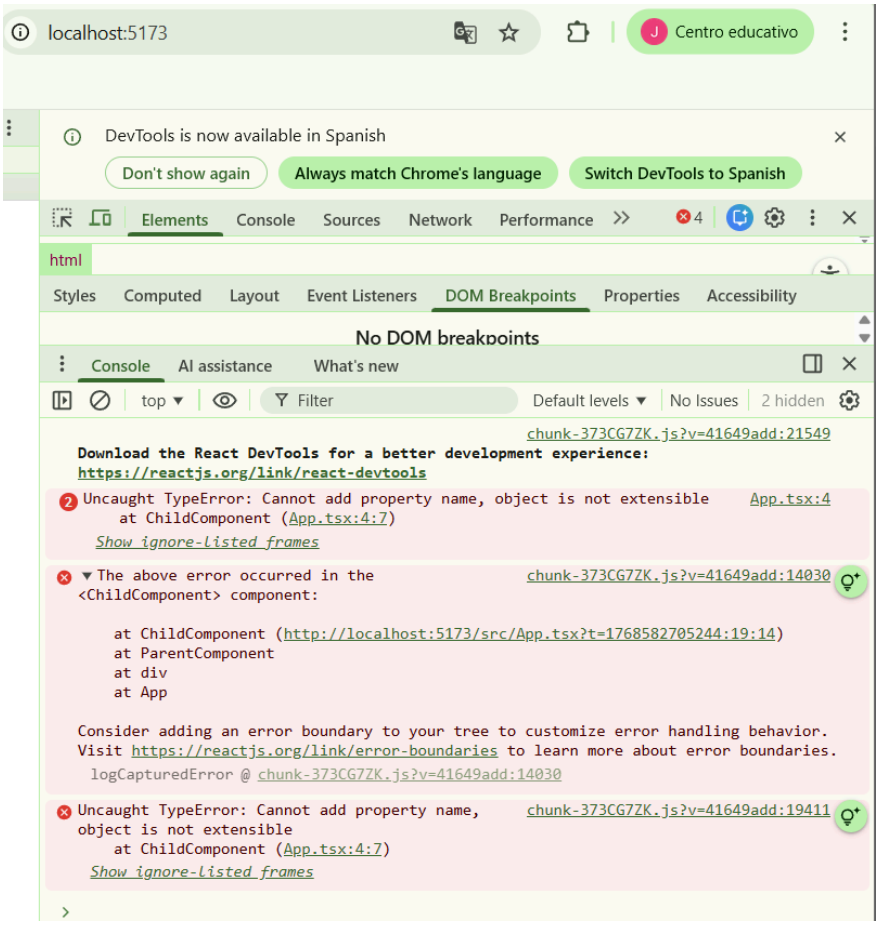
```
1  import './index.css'
2
3  function ChildComponent(props) {
4    return <p>{props.greetings()}</p>
5  }
6
7  function ParentComponent() {
8    function greetings(){
9      return "Frase chorra que ahora mismo no se me ocurre soy el mejor Y TENGO UNA CAMISETA FIRMADA!"
10     }
11     return <ChildComponent
12     greetings ={greetings} />
13 }
14
15 function App() {
16   return (
17     <div>
18       <ParentComponent />
19     </div>
20   )
21 }
22
23 export default App
24
```

(Ilustración 5. Captura de pantalla del código donde definimos la función en el padre y la pasamos como props, llamándola desde el hijo. Elaboración propia.)



(Ilustración 6. Captura de pantalla de la aplicación funcionando correctamente haciendo la clase padre una función y llamándola desde el hijo. Elaboración propia.)

PARTE D: Las props son inmutables



(Ilustración 7. Captura de pantalla del error que nos aparece en “Inspeccionar” de nuestra página web. Elaboración propia)

PREGUNTA: ¿Por qué ocurre esto?
El error ocurre porque en `ChildComponent` se intenta añadir o modificar una propiedad (name) en un objeto que no es extensible.

PARTE E: Estado con `useState`

App.tsx > ParentComponent

```tsx
import './index.css'
import { useState } from 'react'

function ChildComponent(props) {
props.name = "Pablo" // NO se permite
return <p>Hello profesora (guiño guiño)! my name is
{props.name}</p>
}

function ParentComponent() {
  const [name, setName] = useState("Juanaco")
  return (
  <>
    <h1>Hola chavales {name}</h1>
    <button onClick={() => setName("Juanky")}>Cambiar nombre de los DOBLEJ</button>
  </>
  )
}

function App() {
  return (
    <div>
      <ParentComponent />
    </div>
  )
}

export default App
```

(Ilustración 8. Captura de pantalla del código usando `useState` con un botón que cambia el nombre a Juanky en evento onClick. Elaboración propia.)

(Ilustración 9. Captura de pantalla de la aplicación mostrando el botón para cambiar el nombre, y como efectivamente cambia. Elaboración propia.)

PARTE F: Pasar estado y `setState` a un componente hijo

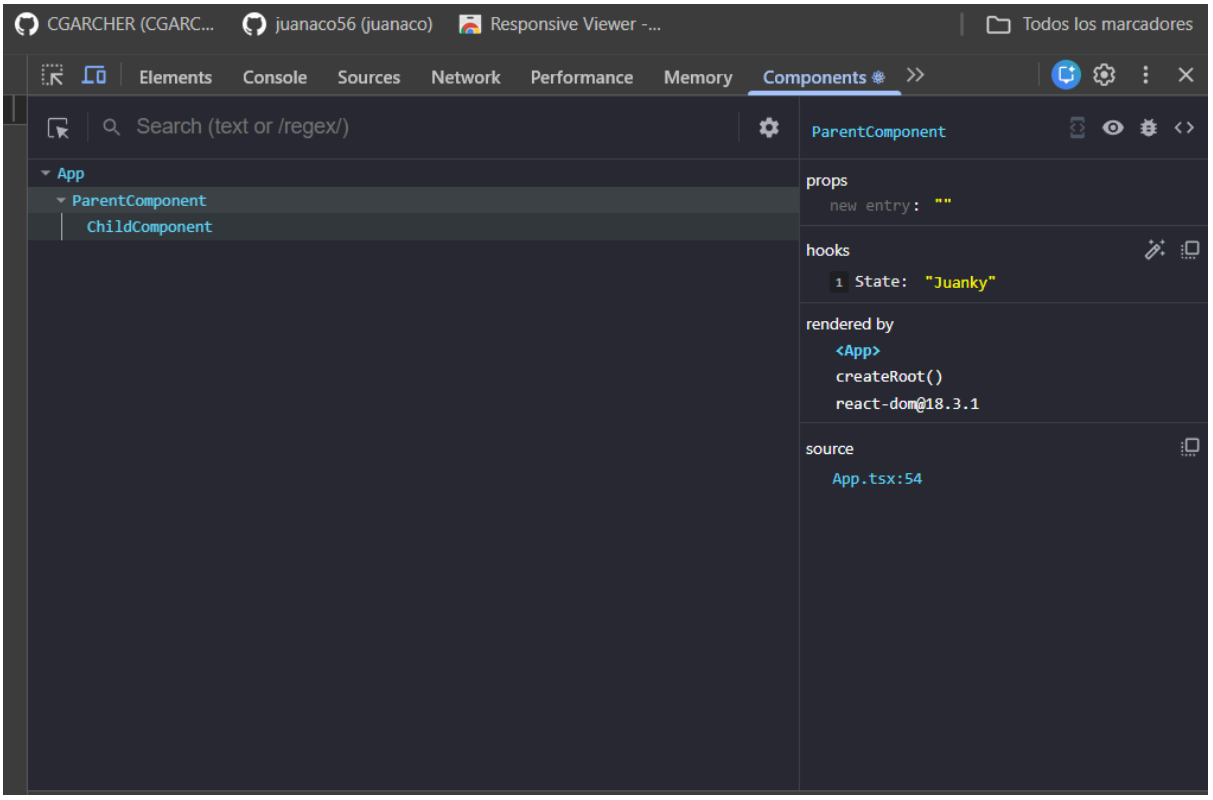
App.tsx > ...

```jsx
import './index.css'
import { useState } from 'react'

function ChildComponent(props) {
  return (
    <>
      <h1>Hello {props.name}</h1>
      <button onClick={() => props.setName("Juanky")}>Cambiar a Juanky</button>
      <br/>
      <button onClick={() => props.setName("Pablo")}>Cambiar a Pablo</button>
    </>
  )
}

function ParentComponent() {
  const [name, setName] = useState("Juanaco")
  return <ChildComponent name={name} setName = {setName}/>
}

function App() {
  return (
    <div>
      <ParentComponent />
    </div>
  )
}

export default App
```

(Ilustración 10. Captura de pantalla mostrando el código en el que se muestra como pasamos el estado y la función `setName` al componente hijo, el hijo muestra el nombre y añade un botón, en este caso 2, para cambiarlo. Elaboración propia.)

(Ilustración 11. Captura de pantalla de la aplicación mostrando los 2 botones mencionados previamente y como cambia al darle gracias al evento `onClick`. Elaboración propia.)

PARTE G: Inspeccionar props y state con React DevTools



(Ilustración 12. Captura de pantalla de la aplicación siendo inspeccionada con nuestra extensión: “React Developer Tools”, donde se aprecia el prop y el state. Al darle al botón, el valor de `props.name` cambia. Elaboración propia.)