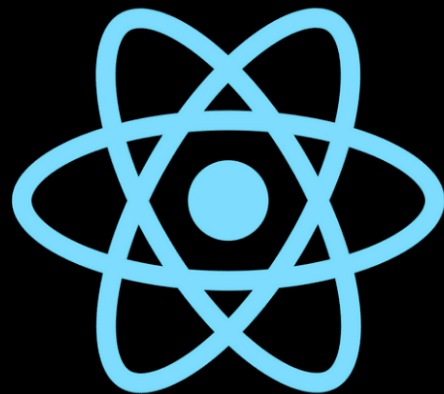


Desarrollo web en entorno cliente - Nivel 4 React

Pablo Valero López

16/01/2026

2ºDAW



React JS

Índice:

[Objetivos](#)

[Parte A: Props básicas \(Padre -> Hijo\)](#)

[Parte B: Pasar múltiples props](#)

[Parte C: Pasar funciones como props](#)

[Parte D: Las props son inmutables](#)

[Parte E: Estado con useState](#)

[Parte F: Pasar estado y setState a un componente hijo](#)

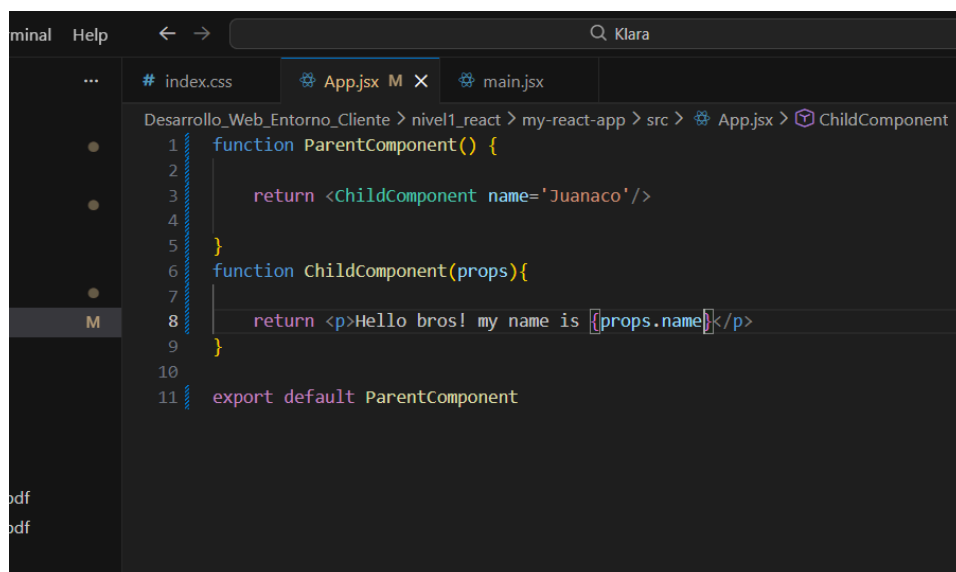
[Parte G: Inspeccionar props y state con React DevTools](#)

Objetivos

- Comprender qué son las props y cómo se pasan de un componente padre a un componente hijo.
- Usar props de distintos tipos: string, number, array y function.
- Entender por qué las props son inmutables.
- Crear y actualizar estado con el hook useState.
- Pasar estado y su función actualizadora (setState) a un componente hijo.
- Inspeccionar props y state usando React Developer Tools (DevTools).

Parte A: Props básicas (Padre -> Hijo)

En la siguiente imagen se muestra el código con el props básico de Juanaco.



```
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > ChildComponent
1  function ParentComponent() {
2
3      return <ChildComponent name='Juanaco' />
4
5  }
6  function ChildComponent(props){
7
8      return <p>Hello bros! my name is {props.name}</p>
9
10 }
11 export default ParentComponent
```

Imagen 1: Código con props básicos. Elaboración propia.

En la siguiente imagen se muestra el resultado.

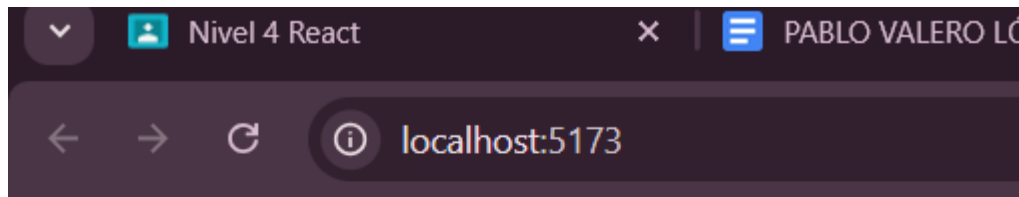


Imagen 2: Resultado de props básicos. Elaboración propia.

Parte B: Pasar múltiples props

En la siguiente imagen se muestra el código con los múltiples props de Iker.



Imagen 3: Código de múltiples props. Elaboración propia.

En la siguiente imagen se muestra el resultado.

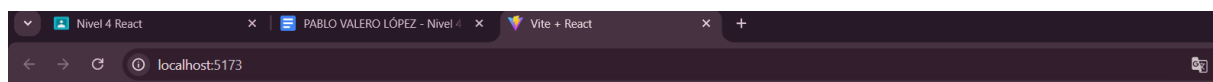
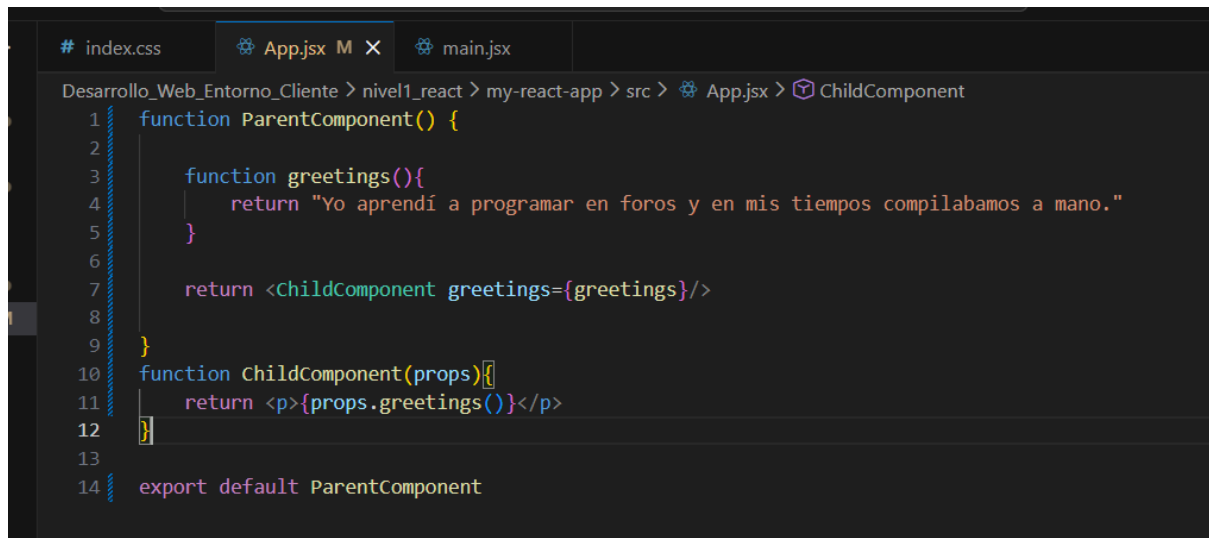


Imagen 4: Resultado de múltiples props. Elaboración propia.

Parte C: Pasar funciones como props

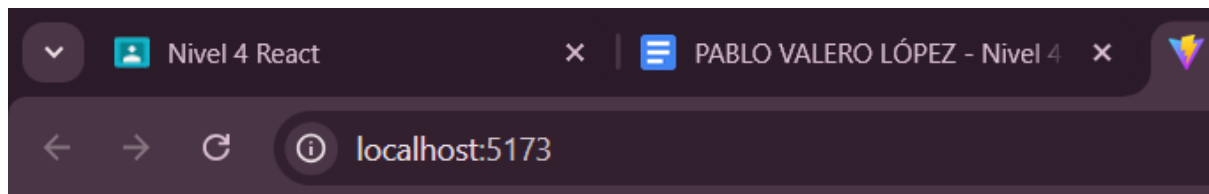
En la siguiente imagen se muestra la función dentro del padre.



```
# index.css App.jsx M X main.jsx
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > ChildComponent
1 function ParentComponent() {
2
3   function greetings(){
4     return "Yo aprendí a programar en foros y en mis tiempos compilabamos a mano."
5   }
6
7   return <ChildComponent greetings={greetings}/>
8
9 }
10 function ChildComponent(props){
11   return <p>{props.greetings()}</p>
12 }
13
14 export default ParentComponent
```

Imagen 5: Código de funciones dentro de la clase padre. Elaboración propia.

En la siguiente imagen se muestra el resultado de la función dentro de padre pasa a hija.



Yo aprendí a programar en foros y en mis tiempos compilabamos a mano.

Imagen 6: Resultado de funciones dentro de la clase padre. Elaboración propia.

Parte D: Las props son inmutables

En la siguiente imagen se muestra el código de los props y como estos son inmutables.

```
... # index.css App.jsx M X main.jsx
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > ChildComponent
1 function ParentComponent() {
2
3   function greetings(){
4     return "Yo aprendí a programar en foros y en mis tiempos compilabamos a mano."
5   }
6
7   return <ChildComponent greetings={greetings}/>
8
9 }
10 function ChildComponent(props) {
11   props.name = "Pablo"
12   return <p>Hello profesora (guiño guiño)! my name is
13     {props.name}</p>
14 }
15
16 export default ParentComponent
```

Imagen 7: Código de props inmutables. Elaboración propia.

En la siguiente imagen se muestra el resultado.

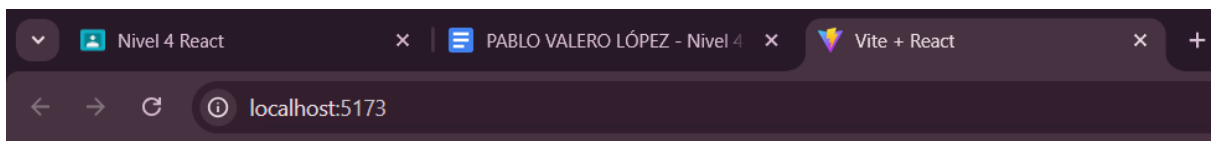
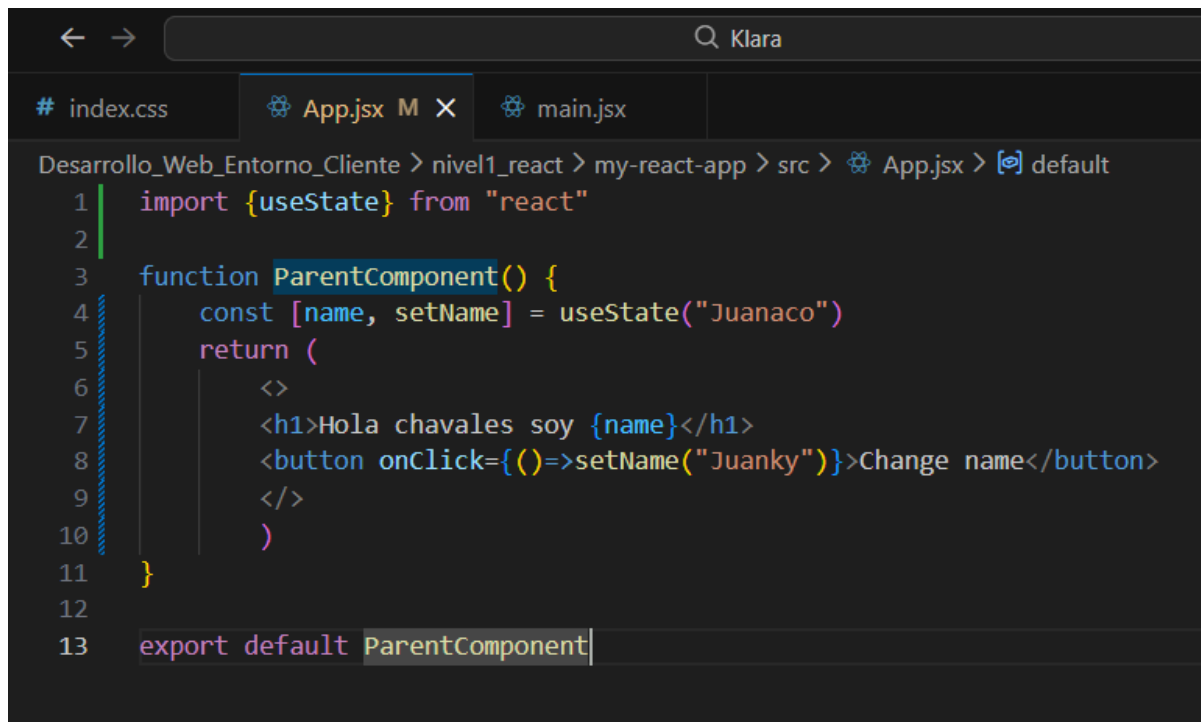


Imagen 8: Resultado de props inmutables. Elaboración propia.

Parte E: Estado con useState

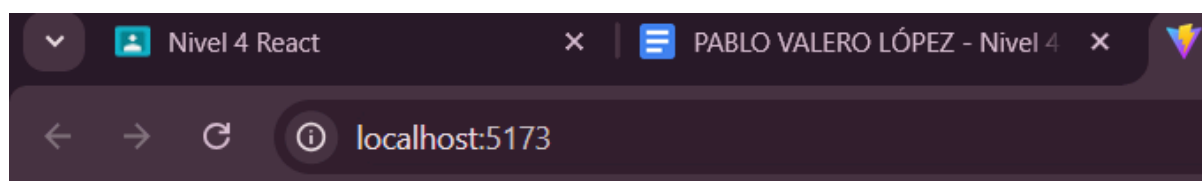
En la siguiente imagen se muestra el código de useState.



```
1 import {useState} from "react"
2
3 function ParentComponent() {
4   const [name, setName] = useState("Juanaco")
5   return (
6     <>
7       <h1>Hola chavales soy {name}</h1>
8       <button onClick={()=>setName("Juanky")}>Change name</button>
9     </>
10   )
11 }
12
13 export default ParentComponent
```

Imagen 9: Código de useState. Elaboración propia.

En la siguiente imagen se muestra el resultado.



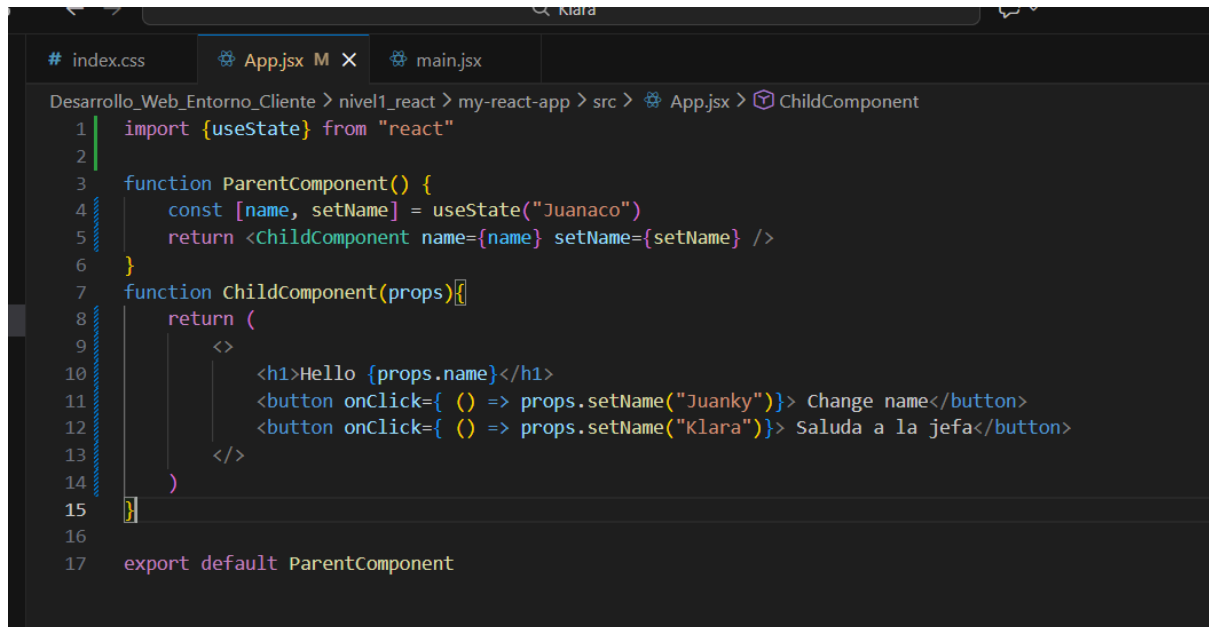
Hola chavales soy Juanaco

Change name

Imagen 10: Resultado de useState. Elaboración propia.

Parte F: Pasar estado y setState a un componente hijo

En la segunda imagen se muestra el código con los botones del cambio de nombre.



```
# index.css App.jsx M X main.jsx
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > ChildComponent
1 import {useState} from "react"
2
3 function ParentComponent() {
4   const [name, setName] = useState("Juanaco")
5   return <ChildComponent name={name} setName={setName} />
6 }
7 function ChildComponent(props){
8   return (
9     <>
10      <h1>Hello {props.name}</h1>
11      <button onClick={ () => props.setName("Juanky")}> Change name</button>
12      <button onClick={ () => props.setName("Klara")}> Saluda a la jefa</button>
13    </>
14  )
15 }
16
17 export default ParentComponent
```

Imagen 11: Código de setNameElaboración propia.

En la siguiente imagen se muestra el resultado del nuevo botón “Saluda a la jefa”.

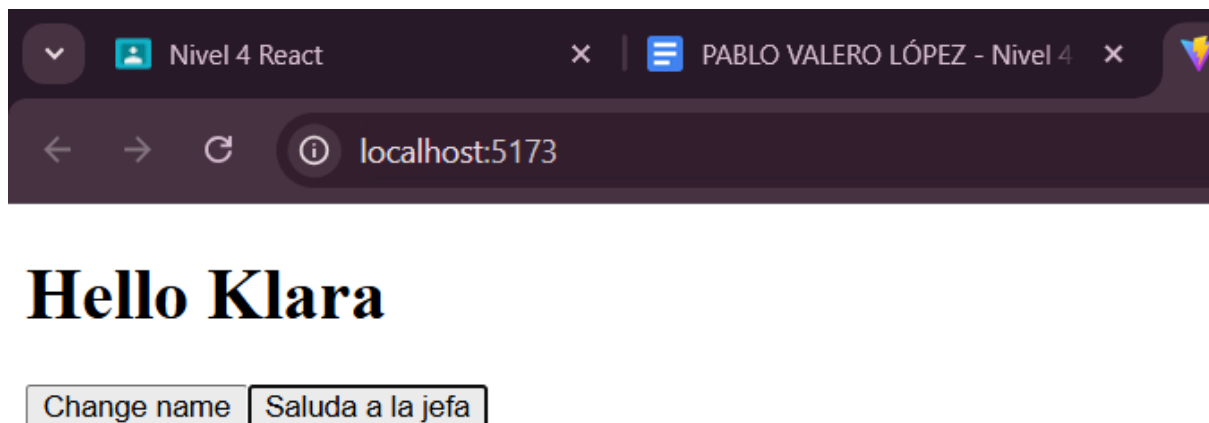


Imagen 12: Resultado de saludar a la jefa. Elaboración propia.

Parte G: Inspeccionar props y state con React DevTools

En la siguiente imagen se muestra os props y state con React DevTools

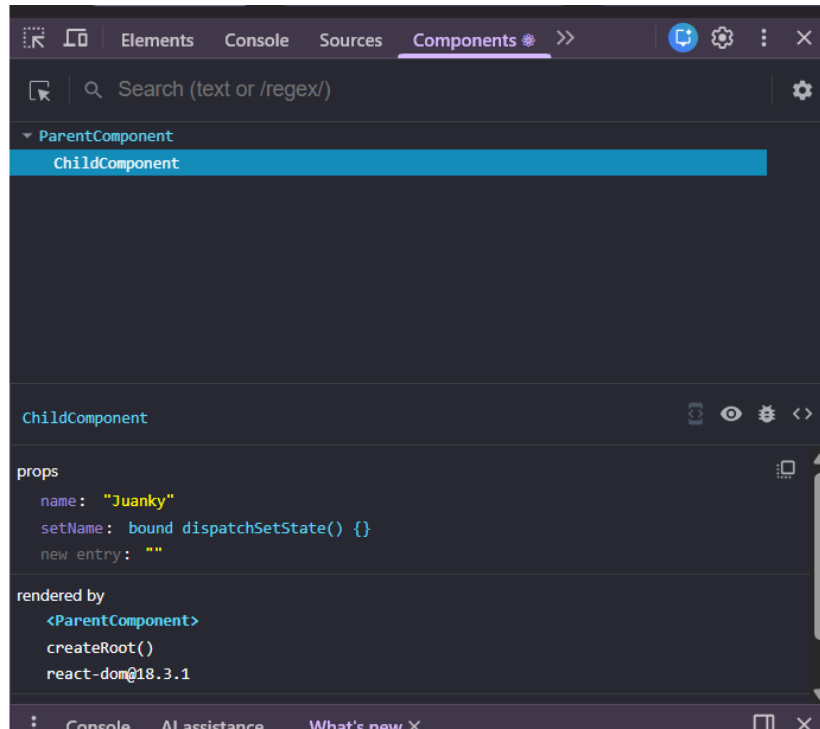


Imagen 13: Resultado de pulsar el primer botón. Elaboración propia.

En la siguiente imagen se muestra el resultado de pulsar el botón “Saludar a la jefa”.

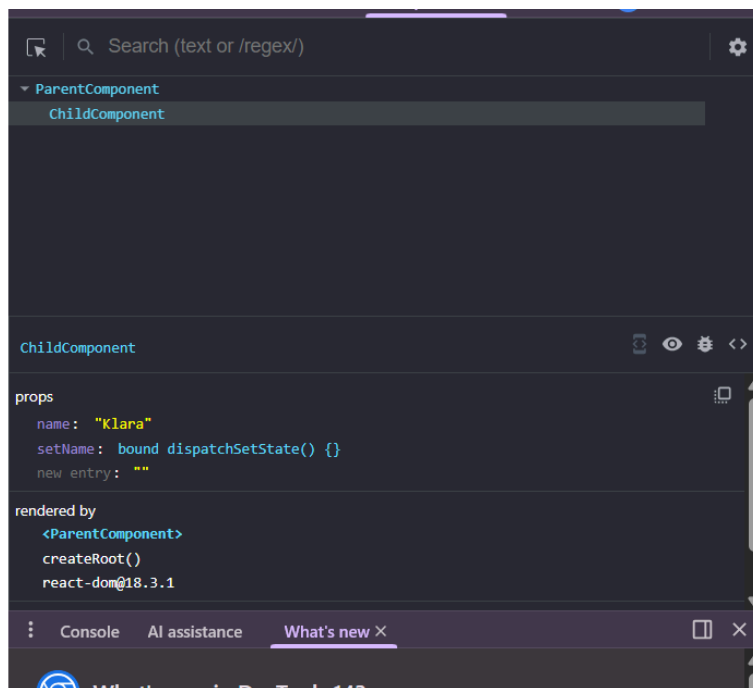


Imagen 14: Resultado de saludar a la jefa. Elaboración propia.