

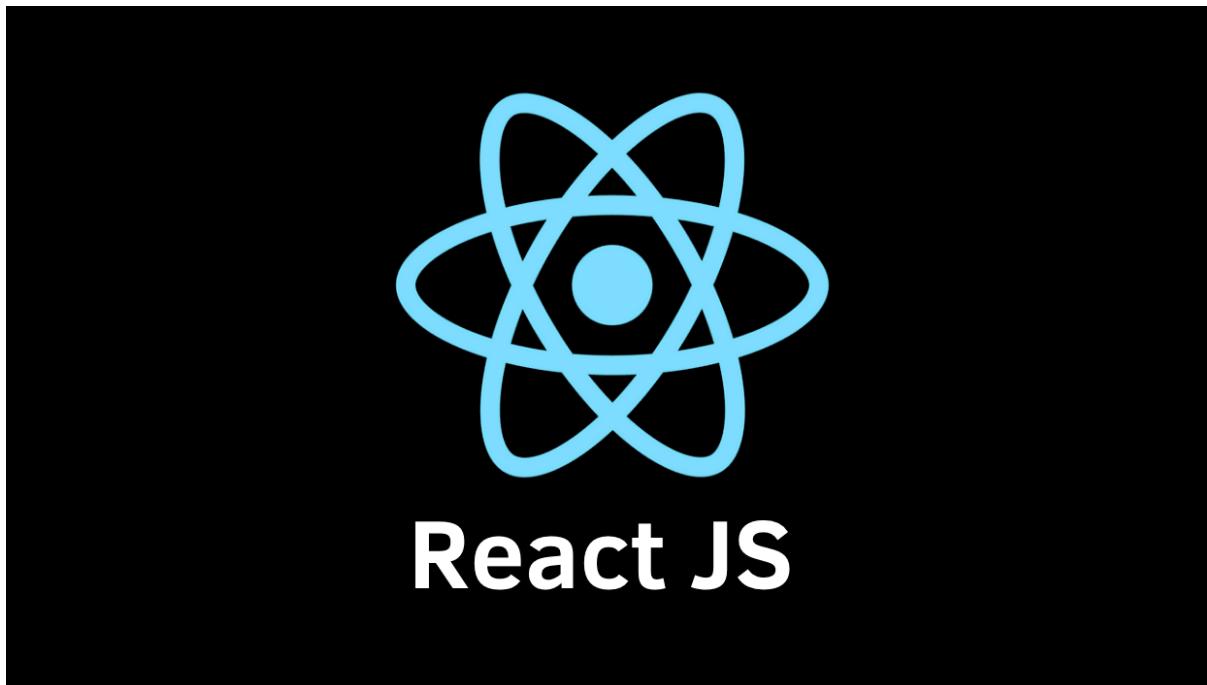
Desarrollo web en entorno cliente -

Nivel 3 React

Pablo Valero López

15/01/2026

2ºDAW



Índice:

Objetivos

[Parte A: JSX básico \(guardar y devolver un elemento\)](#)

[Parte B: Expresiones dentro de JSX con {}](#)

[Parte C: Renderizar una lista con map\(\) + key](#)

[Parte D: className y estilos rápidos](#)

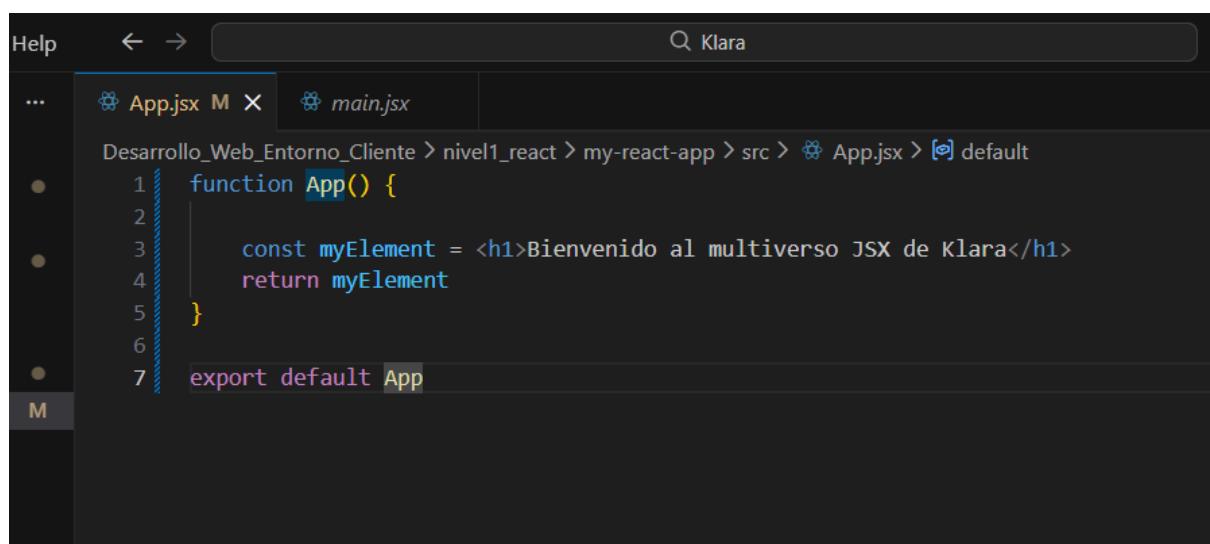
[Preguntas extra para el PDF](#)

Objetivos

- Entender qué es JSX y por qué se usa en React.
- Guardar JSX en variables y devolverlo desde un componente.
- Insertar expresiones JavaScript dentro de JSX con llaves {}.
- Renderizar listas con map() y usar la prop key correctamente.
- Aplicar clases CSS usando className.

Parte A: JSX básico (guardar y devolver un elemento)

En la primera imagen se muestra el código con el elemento creado.



The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with 'Help' and back/forward buttons. A search bar contains the text 'Klara'. Below the navigation bar, there are two tabs: 'App.jsx M X' and 'main.jsx'. The 'App.jsx' tab is active. The code editor displays the following code:

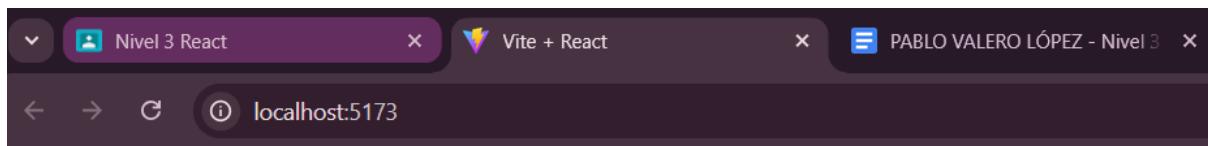
```
function App() {
  const myElement = <h1>Bienvenido al multiverso JSX de Klara</h1>
  return myElement
}

export default App
```

The code is written in JSX, defining a function component named 'App' that returns an `<h1>` element with the text 'Bienvenido al multiverso JSX de Klara'. The code editor also shows a file tree on the left side, indicating the project structure: 'Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx'.

Imagen 1: Crear elemento JSX. Elaboración propia.

En la siguiente imagen se muestra el resultado del elemento JSX.



Bienvenido al multiverso JSX de Klara

Imagen 2: Resultado de elemento JSX. Elaboración propia.

Parte B: Expresiones dentro de JSX con {}

En la siguiente imagen se muestra la clase con el color dentro del archivo index.css.

A screenshot of a code editor showing a file named "index.css". The code contains a single CSS rule: ".text-lowercase{ color: palevioletred; }". The code editor interface includes tabs for "index.css" and "main.jsx", and a status bar showing the file path: "Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app >".

Imagen 3: La clase creada dentro de index.css. Elaboración propia.

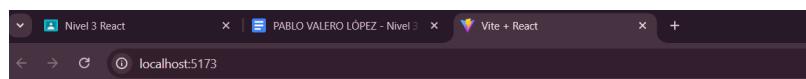
En la siguiente imagen se muestra el código con el texto y el cambio de la clase.

The screenshot shows a code editor window with three tabs at the top: index.css (marked with a 'U'), App.jsx (marked with an 'M' and has a blue underline), and main.jsx (marked with an 'M'). The current view is the App.jsx tab, which contains the following code:

```
1 function App() {  
2     const cambiarClase= 'text-lowercase'  
3     const texto = 'Hola, humano esclavizado de DAW2. React detectado check'  
4     return <h1 className={cambiarClase}>{texto}</h1>  
5 }  
6  
7 export default App
```

Imagen 4:Código con la referencia de la clase. Elaboración propia.

En la siguiente imagen se muestra el resultado de las dos imágenes anteriores.



Hola, humano esclavizado de DAW2. React detectado check

Imagen 5: Resultado de cambio de clase. Elaboración propia.

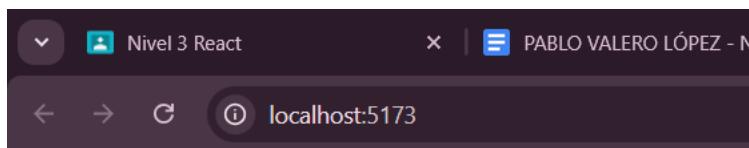
Parte C: Renderizar una lista con map() + key

En la siguiente imagen se muestra en map con la función que los muestra.

```
# index.css U   ⚡ App.js M X   ⚡ main.jsx M
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > ⚡ App.js > [d] default
1  function App() {
2
3      const users = [
4          {id:1, name:'Jose', role:'Web Developer'},
5          {id: 2, name:'Estefania', role: 'Web Developer'},
6          {id: 3, name:'Ruben', role: 'Team Leader'},
7          {id: 4, name:'Juanky', role: 'Web Developer'}
8      ]
9      return (
10         <>
11         <p>Lista de usuarios activos: </p>
12         <ul>
13             {users.map(function (users){
14                 return (
15                     <li key={users.id}>
16                         {users.name} — {users.role}
17                     </li>
18                 )
19             })}
20         </ul>
21     
22 }
23
24
25 export default App
```

Imagen 6: Map con la lista de usuarios. Elaboración propia.

En la siguiente imagen se muestra el resultado.



Lista de usuarios activos:

- Jose — Web Developer
- Estefania — Web Developer
- Ruben — Team Leader
- Juanky — Web Developer

Imagen 7: Resultado del map con los usuarios. Elaboración propia.

Parte D: className y estilos rápidos

En la siguiente imagen se muestra el map con el cambio de la className para poder cambiar el color dependiendo del role.

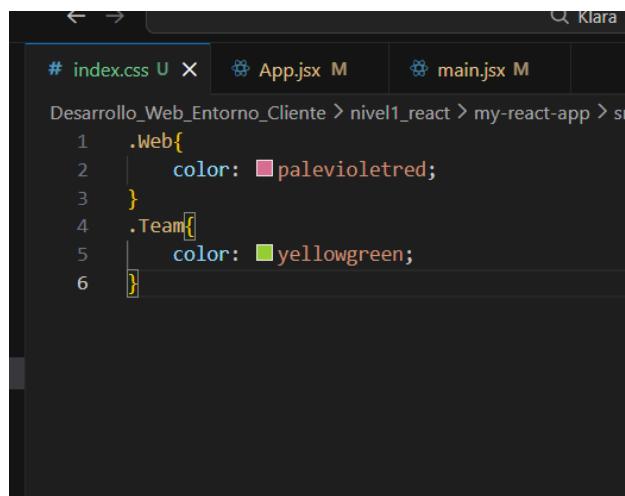
```
rrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > App.jsx > App > users.map
function App() {

    const users = [
        {id:1, name:'Jose', role:'Web Developer'},
        {id: 2, name:'Estefania', role:'Web Developer'},
        {id: 3, name:'Ruben', role:'Team Leader'},
        {id: 4, name:'Juanky', role:'Web Developer'}
    ]
    return (
        <>
        <p>Lista de usuarios activos:</p>
        <ul>
            {users.map(function (users){
                return (
                    <li key={users.id} className={users.role}>
                        {users.name} - {users.role}
                    </li>
                )
            })}
        </ul>
    )
}

export default App
```

Imagen 8: Map con la lista de usuarios + class. Elaboración propia.

En la siguiente imagen se muestra la clase con los colores diferenciados.

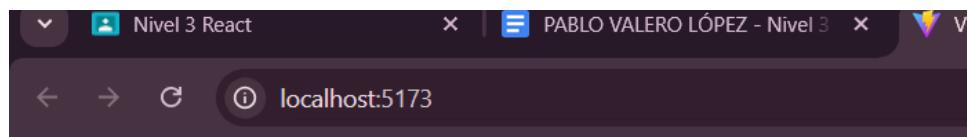


The screenshot shows a code editor window with three tabs at the top: '# index.css U X', 'App.jsx M' (which is the active tab), and 'main.jsx M'. The content of the '# index.css' tab is as follows:

```
Desarrollo_Web_Entorno_Cliente > nivel1_react > my-react-app > src > index.css
1 .Web{
2     color: #palevioletred;
3 }
4 .Team{
5     color: #yellowgreen;
6 }
```

Imagen 9: Clases con colores diferenciados. Elaboración propia.

En la siguiente imagen se muestra el resultado.



Lista de usuarios activos:

- Jose — Web Developer
- Estefania — Web Developer
- Ruben — Team Leader
- Juanky — Web Developer

Imagen 10: Resultado del map con las className. Elaboración propia.

Preguntas extra para el PDF

1. ¿Qué es JSX con tus palabras?

Es una extensión de la sintaxis de JavaScript que permite escribir como si fuera **HTML** pero viene de archivos **.js** o **.jsx**.

2. ¿Por qué usamos {} dentro de JSX? Pon un ejemplo.

Las llaves se utilizan para decirle a React que lo que hay dentro de las mismas no es **HTML** sino **JavaScript**. Ejemplo: `<h1>2+2<h1/>` = **2+2**. `<h1>{2+2}<h1/>` = **4**. En la primera React lo **interpreta como texto** y muestra lo mismo, en el segundo caso, **JavaScript realiza la suma** y muestra el resultado.

3. ¿Para qué sirve la prop key en listas?

El elemento prop key sirve para que React diferencie los elementos en las listas.

4. ¿Por qué usamos className y no class en React?

Utilizamos la **className en React** porque **class** es una **palabra reservada de JavaScript**.