



Componentes, Props y Eventos en React

Centro de Servicios y Gestión Empresarial
SENA Regional Antioquia



www.sena.edu.co

Concepto – Componentes, Props y Eventos

En React, el desarrollo de aplicaciones se organiza en torno a **componentes, estados, props y eventos**. Estos conceptos son la base para crear interfaces dinámicas y reutilizables.

- Los **componentes** encapsulan partes de la interfaz.
- Las **props** permiten que un componente reciba datos desde sus padres, manteniéndolo flexible y reutilizable.
- El **estado** es un sistema de datos mutable interno que ayuda a que el componente reaccione a los cambios.
- Los **eventos** permiten que los usuarios interactúen con los componentes y desencadenen cambios en el estado o acciones en la aplicación.

The React logo, a stylized white atom with three elliptical orbits and a central dot, is positioned to the left of the word "Componentes".

Componentes

Componentes

Un **componente** en React es una pieza independiente y reutilizable de código que representa una parte de la interfaz de usuario.

Los componentes permiten **descomponer la interfaz en partes más pequeñas y manejables**, lo que facilita el desarrollo, mantenimiento y reutilización del código. Cada componente puede tener su propio estado y propiedades (props), y puede incluir lógica para manejar eventos y renderizar diferentes partes de la interfaz.



Componentes - Sintaxis

```
import React from 'react';

function MyComponent() {
  return (
    // Etiquetas HTML
    <p></p>
  );
}

export default MyComponent;
```

- **Importación:** Se importa React desde el paquete react.
- **Declaración:** Se define una función llamada MyComponent.
- **JSX:** La función retorna JSX que describe la interfaz.
- **Exportación:** El componente se exporta para que pueda ser utilizado en otros archivos

Componentes - Sintaxis



```
import React from 'react';

export function ButtonLike() {
  return (
    <button>Me gusta</button>
  );
}
```

01

```
import React from 'react';

function ButtonDislike() {
  return (
    <button>No me gusta</button>
  );
}

export default ButtonDislike;
```

02

```
import React from 'react';

const OtherButton = () => {
  return (
    <button>Other Button</button>
  );
}

export { OtherButton };
```

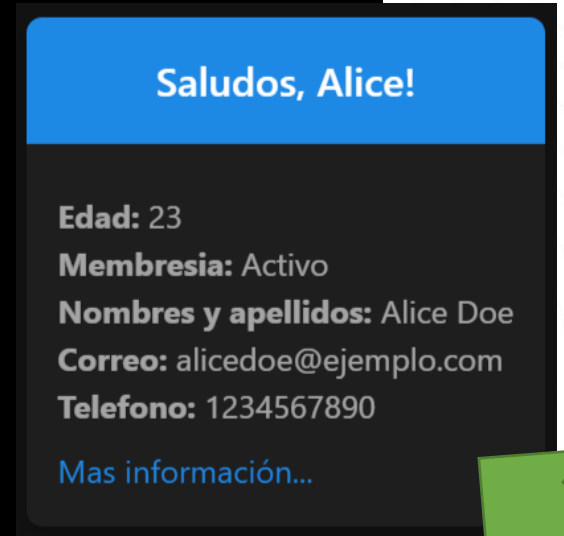
03

Los componentes se pueden componer, es decir, puedes combinarlos para construir interfaces complejas al encapsular y reutilizar funcionalidad en diferentes partes de la aplicación.

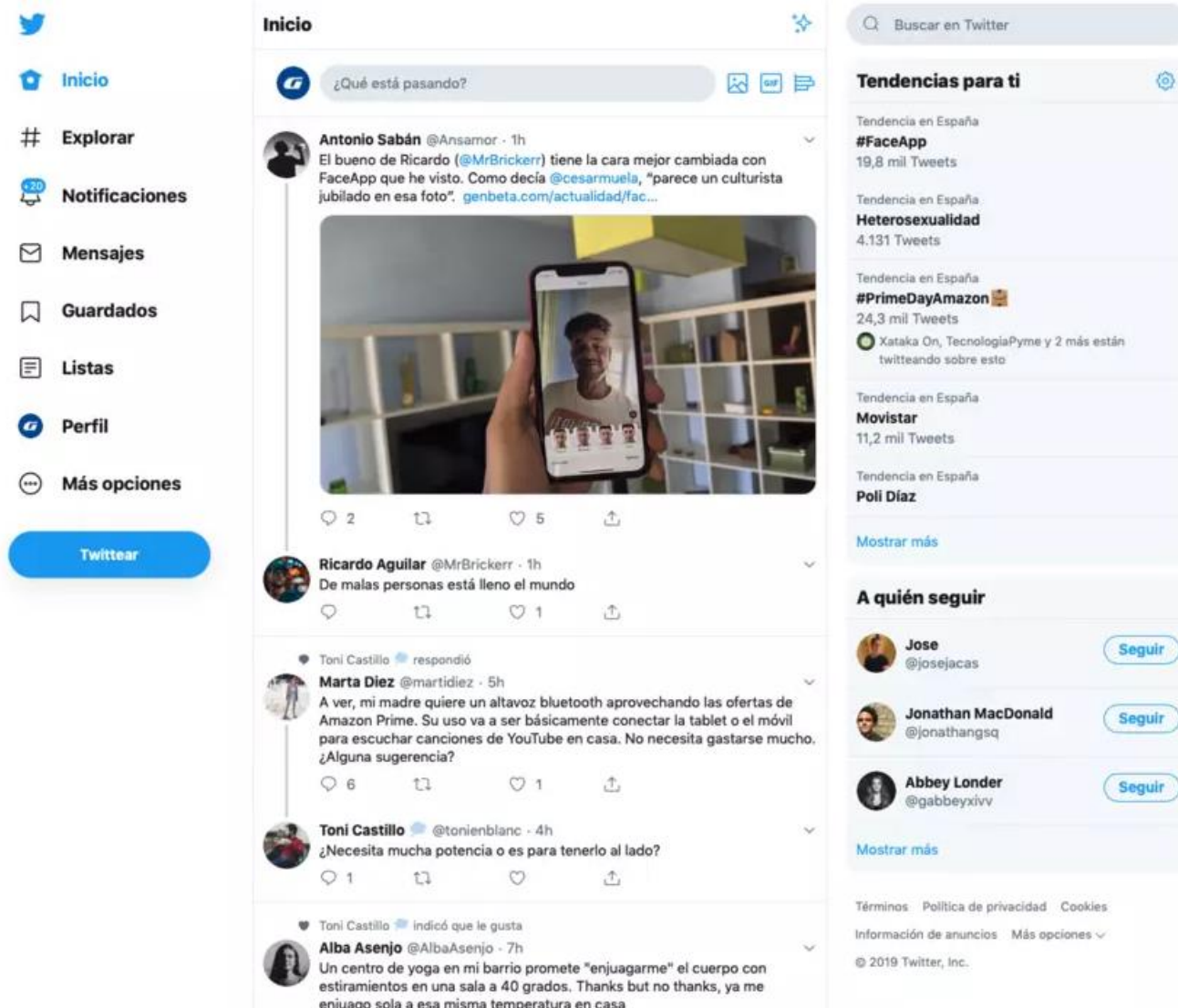
Componentes - Ejemplo



```
export function CardInfo() {  
  return (  
    <div className="card">  
      <div className="card-header">  
        <h5 className="card-title">Saludos, Alice!</h5>  
      </div>  
      <div className="card-body">  
        <p className="card-text">  
          <strong>Edad:</strong> 23 <br />  
          <strong>Membresia:</strong> Activo <br />  
          <strong>Nombres y apellidos:</strong> Alice Doe <br />  
          <strong>Correo:</strong> alicedoe@ejemplo.com <br />  
          <strong>Telefono:</strong> 1234567890 <br />  
        </p>  
        <a href="#" className="card-link">Mas información...</a>  
      </div>  
    </div>  
  );  
}
```



Componentes - Ejercicio



<https://unavatar.io/>



Componentes - Ejercicio



Productos Destacados

300 x 200

Zapatillas de Running
Precio: \$99.99

[Añadir al carrito](#)

300 x 200

Balón de Fútbol
Precio: \$29.99

[Añadir al carrito](#)

300 x 200

Raqueta de Tenis
Precio: \$159.99

[Añadir al carrito](#)

300 x 200

Mancuernas (par)
Precio: \$49.99

[Añadir al carrito](#)

300 x 200

Camiseta Deportiva
Precio: \$24.99

[Añadir al carrito](#)

300 x 200

Bicicleta de Montaña
Precio: \$599.99

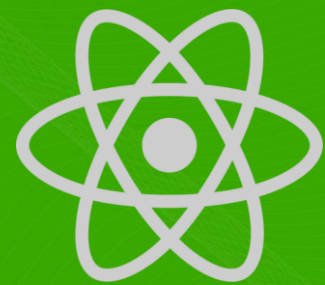
[Añadir al carrito](#)

300 x 200

Zapatillas de Running
Precio: \$99.99

[Añadir al carrito](#)

A large, thick green arrow originates from the bottom left of the image and points towards the "Zapatillas de Running" product card, which is highlighted with a white border and a subtle drop shadow.



Props (Propiedades)

Props

Las **props** son datos que se pasan a los componentes desde sus componentes padres. Son inmutables dentro del componente que las recibe, lo que significa que no pueden cambiarse desde el propio componente.


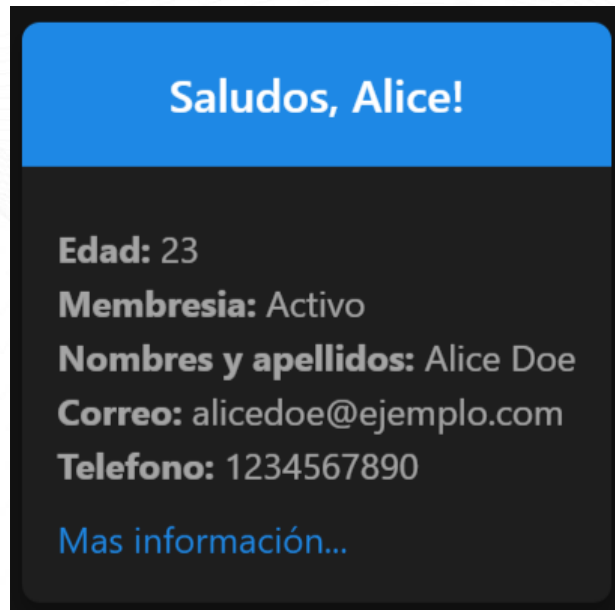
```
function Greeting({ name }) {  
  return <h1>Hello, {name}!</h1>;  
}  
  
function App() {  
  return <Greeting name="Alice" />;  
}
```

En este ejemplo, el componente Greeting recibe name como una prop desde App. Las props permiten que un componente sea más flexible y reutilizable, ya que puede recibir diferentes valores en diferentes contextos.

Props - Ejemplo



```
// Datos de usuario
const usuario = {
  nick: 'Alice',
  edad: 23,
  membresia: 'Activo',
  nombres: 'Alice Doe',
  correo: 'alicedoe@ejemplo.com',
  telefono: '1234567890'};
```

A large green curved arrow points from the user data code block to the component code block.

```
// Componente principal - Padre
export function App() {
  return (
    <SaludosComp
      nick={usuario.nick}
      edad={usuario.edad}
      membresia={usuario.membresia}
      nombres={usuario.nombres}
      correo={usuario.correo}
      telefono={usuario.telefono}/>
  );
}
```

Props - Ejemplo



```
export function SaludosComp({nick, edad, membresia, nombres, correo, telefono}) {  
  return (  
    <div className="card">  
      <div className="card-header">  
        <h5 className="card-title">Saludos, {nick}!</h5>  
      </div>  
      <div className="card-body">  
        <p className="card-text">  
          <strong>Edad:</strong> {edad} <br />  
          <strong>Membresia:</strong> {membresia} <br />  
          <strong>Nombres y apellidos:</strong> {nombres} <br />  
          <strong>Correo:</strong> {correo} <br />  
          <strong>Telefono:</strong> {telefono} <br />  
        </p>  
        <a href="#" className="card-link">Mas información...</a>  
      </div>  
    </div>  
  );  
}
```

Saludos, Alice!

Edad: 23

Membresia: Activo

Nombres y apellidos: Alice Doe

Correo: alicedoe@ejemplo.com

Telefono: 1234567890

[Mas información...](#)

Props - Ejercicios



Twitter / X Follow Card



SENA Comunica
@senaComunica

Seguir



Alvaro Pérez Niño
@majash29

Siguiendo



SENA Comunica
@senaComunica

Siguiendo



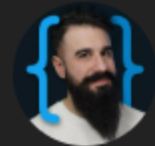
Alvaro Pérez Niño
@majash29

Seguir



Miguel Ángel Durán
@midudev

Siguiendo



Brais Moure
@mouredev

Seguir

Props - Ejercicios



Productos Destacados

<p>300 x 200</p> <p>Zapatillas de Running</p> <p>Precio: \$99.99</p> <p>Eliminar del carrito</p>	<p>300 x 200</p> <p>Balón de Fútbol</p> <p>Precio: \$29.99</p> <p>Añadir al carrito</p>	<p>300 x 200</p> <p>Raqueta de Tenis</p> <p>Precio: \$159.99</p> <p>Eliminar del carrito</p>
<p>300 x 200</p> <p>Mancuernas (par)</p> <p>Precio: \$49.99</p> <p>Añadir al carrito</p>	<p>300 x 200</p> <p>Camiseta Deportiva</p> <p>Precio: \$24.99</p> <p>Eliminar del carrito</p>	<p>300 x 200</p> <p>Bicicleta de Montaña</p> <p>Precio: \$599.99</p> <p>Añadir al carrito</p>



Estados

Estados

El **estado** es una forma de almacenar y gestionar datos dentro de un componente. A diferencia de las **props**, el estado es **mutable**: el componente puede actualizar su propio estado, lo que permite que la interfaz reaccione a los cambios en el estado interno del componente.

Para componentes funcionales, el estado se maneja con el Hook `useState`.

```
// 01 Importar de React - useState
import React, { useState } from 'react';

// 02 Sintaxis del useState
// Elemento, set del elemento = Estado inicial
const [name, setName] = useState('');
```

En este ejemplo, el estado **name** se inicializa en `' '` y se actualiza mediante **setName**. Cuando el usuario hace clic en el botón, se actualiza el estado y React vuelve a renderizar el componente con el nuevo valor de **name**.



Eventos

Eventos

Los **eventos** son acciones que los usuarios o el sistema pueden realizar, como **hacer clic en un botón** o **presionar una tecla**. En React, los eventos se manejan mediante atributos de eventos, como **onClick** para clics de botones, **onChange** para entradas de texto. Estos eventos se pasan en **camelCase** y reciben funciones como valores.

```
import React from 'react';

export function ClickButton() {
  const handleClick = () => alert('Click en el botón');
  return(
    <button onClick={handleClick}>
      Click!
    </button>
  );}
```

Ejemplo de estados y eventos - onClick



```
import React, { useState } from 'react'
import './App.css'

export function CompOnClick() {
  const [count, setCount] = useState(0)
  const countUp = () => setCount((count) => count + 1)
  return (
    <div className="card">
      <div className="card-header">
        <h5 className="card-title">Ejemplo No.01</h5>
      </div>
      <div className="card-body">
        <p>Ejemplo de estados y eventos - onClick</p>
        <button onClick={countUp}>
          Click No: {count}
        </button>
      </div>
    </div>
  )
}
```



BtnClick.jsx



```
import React, { useState } from 'react';
import './App.css'
export function CompOnChange() {
  const [inputValueOne, setInputValueOne] = useState(null);
  const [inputValueTwo, setInputValueTwo] = useState(null);
  const handleInputOne = (e) => setInputValueOne(Number(e.target.value));
  const handleInputTwo = (e) => setInputValueTwo(Number(e.target.value));
  const sum = inputValueOne + inputValueTwo;
  return (
    <div className="card">
      <div className="card-header">
        <h5 className="card-title">Ejemplo No.02</h5>
      </div>
      <div className="card-body">
        <p>Ejemplo de estados y eventos - onChange</p>
        <input type="number" value={inputValueOne} onChange={handleInputOne}
          size="5" placeholder="Primer número ..." /> +
        <input type="number" value={inputValueTwo} onChange={handleInputTwo}
          placeholder="Primer número ..." />
        <br /><strong><label> Resultado es: {sum} </label></strong>
      </div></div>
    );
  }
```

Ejemplo de estados y eventos - onChange



OnChange.jsx

Ejemplo de estados y eventos



```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import { CompOnClick } from './BtnClick.jsx'
import { CompOnChange } from './OnChange.jsx'

createRoot(document.getElementById('root')).render(
  <>
    <CompOnClick />
    <CompOnChange />
  </>,
)
```



main.jsx



GRACIAS

Presentó: Alvaro Pérez Niño
Instructor Técnico

Correo: aperezn@misena.edu.co

<http://centrodeserviciosygestionempresarial.blogspot.com/>

Línea de atención al ciudadano: 01 8000 910270

Línea de atención al empresario: 01 8000 910682



@SENAComunica

www.sena.edu.co