🔁 1. Props (short for "Properties")

💡 What are Props?

Props are **read-only** values passed **from parent to child components**. Think of them as **function arguments**.

📦 Example:

jsx Copy Edit

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

🧠 `props.name` here is `"Alice"` — passed from `<App />` into `<Welcome />`.

✅ Props Key Points:

- Props are **immutable** (you can't change them inside the child).
- Help make components **reusable**.
- You can pass **strings, numbers, functions, arrays, objects**, even other components.

🔄 2. State

💡 What is State?

State is a way to store **data that changes over time** — like form inputs, toggles, counters, etc.

In **functional components**, we manage state using the `useState` hook.

📦 Example:

jsx Copy Edit

```jsx
import React, { useState } from 'react'; function Counter() { const [count, setCount] = useState(0); // count = state, setCount = update function return ( <div> <p>You clicked {count} times</p> <button onClick={() => setCount(count + 1)}>Click me</button> </div> ); }
```

✅ State Key Points:

- Managed **within a component**.
- **useState(initialValue)** returns:
  - The current state value.
  - A function to update it.
- Updating state **re-renders the component**.

🔁 State vs Props – Comparison

Feature	Props	State
Passed from	Parent component	Managed inside the component
Read/Write	Read-only	Read & Write
Purpose	Pass data into a component	Handle data that changes
Can update it?	❌ No	✅ Yes (with setState / setX)

🧩 Combined Example

jsx

Copy Edit

```jsx
function WelcomeMessage({ name }) { return <h2>Welcome, {name}!</h2>; } function App() { const [username, setUsername] = useState("React Learner"); return ( <div> <WelcomeMessage name=
```

```
{username} /> <button onClick={() => setUsername("JSX Master")}>Change Name</button> </div> ); }
```

- `username` is **state** in `App`.
- `name` is a **prop** passed to `WelcomeMessage`.