Understanding State and Props in React

1. What are Props?

Props (short for 'properties') are read-only inputs passed to a React component from its parent.

They help components communicate with each other by passing data from parent to child.

Props are immutable - they cannot be changed inside the child component.

Example:

```
function Greeting(props) {
  return <h1>Hello, {props.name}</h1>;
}

<Greeting name="Sara" /> // 'name' is passed as a prop
```

2. What is State?

State is a built-in object that holds data or information about the component.

Unlike props, state is managed within the component and can be changed (mutable).

State is used for things like form input, toggling UI, counters, etc.

State changes trigger re-rendering of the component.

Example in class component:

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }
```

```
  render() {
    return <p>Count: {this.state.count}</p>;
  }
}
```

3. Key Differences Between Props and State

Feature	Props	State
Definition	Read-only data passed from parent	Local data managed by the component
Mutability	Immutable	Mutable
Where Defined	In parent component	Inside the component itself
Usage	To configure a component	To control behavior over time
Trigger Re-render	Yes, when changed by parent	Yes, when updated internally
Access	this.props (class) / props (function)	this.state (class only without hooks)

4. Summary

- Use props to pass data into a component from a parent.

- Use state when you want the component to manage and change its own data.

Think of props like arguments to a function and state like variables declared inside a function.

Both are essential to building interactive and dynamic UIs in React.