Props and State

Question 1: What are props in React.js? How are props different from state ?

ANS.

What are props in React.js ? :-

Props (short for *properties*) are read-only inputs passed from a parent component to a child component. They allow components to receive data and behave dynamically.

Props help make components reusable and configurable.

Example :

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


// Using the component
<Greeting name="Uday" />
```

Difference between Props and State :-

Feature	Props	State
Definition	Data passed from parent to child	Data managed inside the component

Mutability	Read-only (cannot be changed by child)	Can be changed using setState / useState
Who controls it	Parent component	The component itself
Purpose	To pass data & configuration	To manage dynamic data
Re-render	Changes in props cause re-render	Changes in state cause re-render

State Example :-

```
import { useState } from "react";


function Counter() {
  const [count, setCount] = useState(0);


  return (
    <button onClick={() => setCount(count + 1)}>
      Count: {count}
    </button>
  );
}
```

Question 2: Explain the concept of state in React and how it is used to manage component data.

Ans.

<u>Concept of State in React ;-</u>

State is a built-in React object used to store and manage data that can change over time within a component.
When state changes, React automatically re-renders the component to reflect the updated data in the UI.

State is mainly used for dynamic and interactive data like:

- Form inputs
- Counters
- Toggle buttons
- API responses
- User interactions

Why do we need State?

JavaScript variables don’t update the UI when their value changes.
State solves this problem by keeping the UI in sync with data changes.

Using State in Functional Components (useState)

```
import { useState } from "react";


function Counter() {

  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>
        Increase
      </button>
    </div>
  );
}
```

count → current state value

setCount → function to update state

useState(0) → initial state value

How State Manages Component Data :-

1. Component renders with initial state
2. User performs an action (click, input, etc.)
3. State update function (setState / setCount) is called
4. React re-renders the component with new state
5. UI updates automatically

State in Class Components :

```
class Counter extends React.Component {
  state = { count: 0 };

  render() {
    return (
      <button onClick={() => this.setState({ count: this.state.count + 1 })}>
        Count: {this.state.count}
      </button>
    );
  }
}
```

Important Rules of State :-

Do not modify state directly :

```
count = count + 1; // wrong
```

Always use the state updater function :

```
setCount(count + 1); // correct
```
