1. Props (Parent to Child Communication)

1.1 What are Props?

- Props = Properties
- Read-only data passed from parent to child
- Helps create **reusable components**

1.2 Simple Example

```
function Child({ name, age }) {
  return <h2>Hello {name}, Age: {age}</h2>;
}

function Parent() {
  return <Child name="Sunil" age={25} />;
}
```

1.3 Default Props

```
Child.defaultProps = { name: 'Guest', age: 18 };
```

1.4 Function as Prop

```
function Child({ greet, name }) {
  return <button onClick={() => greet(name)}>Greet</button>;
}

function Parent() {
  const greet = (name) => alert(`Hello ${name}`);
  return <Child name="Sunil" greet={greet} />;
}
```

2. State (Dynamic Component Data)

- `useState` hook for dynamic data
- State is **local** to the component

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <h2>{count}</h2>
      <button onClick={() => setCount(count + 1)}>+</button>
      <button onClick={() => setCount(count - 1)}>-</button>
    </div>
  );
}
```

---

3. Advanced Example – User Management App (Props + State + CRUD + Event Handling)

3.1 App.js (Parent Component)

```
import React, { useState } from "react";
import UserCard from "./UserCard";
import AddUserForm from "./AddUserForm";

function App() {
  const [users, setUsers] = useState([
    { id: 1, name: "Sunil", age: 25 },
    { id: 2, name: "Aman", age: 30 },
    { id: 3, name: "Rakesh", age: 28 },
  ]);

  const addUser = (user) => {
    setUsers([...users, { ...user, id: Date.now() }]);
  };

  const removeUser = (id) => {
    setUsers(users.filter((user) => user.id !== id));
  };

  const greet = (name) => alert(`Hello ${name}! Welcome!`);

  return (
    <div style={{ padding: "20px" }}>
      <h1>User Management App</h1>
      <AddUserForm addUser={addUser} />
      {users.map((user) => (
        <UserCard
          key={user.id}
```

```
          user={user}
          greet={greet}
          removeUser={removeUser}
        />
      ))}
    </div>
  );
}

export default App;
```

3.2 UserCard.js (Child Component)

```
import React from "react";

function UserCard({ user, greet, removeUser }) {
  return (
    <div style={{ border: "1px solid gray", padding: "10px", marginBottom:
"10px", width: "300px", borderRadius: "5px" }}>
      <h2>{user.name}</h2>
      <p>Age: {user.age}</p>
      <button onClick={() => greet(user.name)}>Greet</button>
      <button onClick={() => removeUser(user.id)} style={{ marginLeft:
"10px" }}>Remove</button>
    </div>
  );
}

UserCard.defaultProps = {
  user: { name: "Guest", age: 18 },
  greet: (name) => alert(`Hello ${name}`),
  removeUser: () => {},
};

export default UserCard;
```

3.3 AddUserForm.js (Child Component)

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

function AddUserForm({ addUser }) {
  const [name, setName] = useState("");
  const [age, setAge] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!name || !age) return alert("Please fill all fields");
    addUser({ name, age: parseInt(age) });
```

```
    setName("");
    setAge("");
  };

  return (
    <form onSubmit={handleSubmit} style={{ marginBottom: "20px" }}>
      <input type="text" placeholder="Name" value={name} onChange={(e) =>
setName(e.target.value)} style={{ marginRight: "10px" }}/>
      <input type="number" placeholder="Age" value={age} onChange={(e) =>
setAge(e.target.value)} style={{ marginRight: "10px" }}/>
      <button type="submit">Add User</button>
    </form>
  );
}

export default AddUserForm;
```

3.4 Features Demonstrated

1. Props (Data and function from parent to child)
2. State management with useState
3. Event handling (click, form submit)
4. CRUD operations (Add, Remove, Read)
5. Default props and reusable components

---

4. Conclusion

- Props = Parent → Child data/function communication
- State = Component's dynamic data
- Event Handling = User interaction → state update
- This example covers **complete practical usage** of Props, State, and Event Handling in React.