# 📘 Week 1 – React Basics

## Day 3: Props – Passing Data into Components

### 🔹 What are Props?

* **Props** (short for *properties*) are a way to pass data from a **parent component** to a **child component**.
* Think of props like **function parameters** in JavaScript.
* They make components **reusable** and **dynamic**.

👉 Without props, every component would be hardcoded and static.

### 🔹 Why do we use Props?

1. **Reusability** – Instead of writing separate components for each case, you can pass different data.
2. **Dynamic Content** – A component can change its behavior or display depending on the data it receives.
3. **Parent-to-Child Communication** – Props are how parents talk to their children in React.

### 🔹 How to Use Props?

1. **Passing Props**

// Parent Component  
function App() {  
 return (  
 <div>  
 <Greeting name="Rehan" />  
 <Greeting name="Boss" />  
 </div>  
 );  
}

1. **Receiving Props**

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

✅ Output:

Hello, Rehan!  
Hello, Boss!

### 🔹 Destructuring Props

Instead of writing props.name, you can **destructure**:

function Greeting({ name }) {  
 return <h1>Hello, {name}!</h1>;  
}

### 🔹 Multiple Props Example

function UserCard({ name, age, city }) {  
 return (  
 <div>  
 <h2>{name}</h2>  
 <p>Age: {age}</p>  
 <p>City: {city}</p>  
 </div>  
 );  
}  
  
function App() {  
 return (  
 <div>  
 <UserCard name="Rehan" age={21} city="Islamabad" />  
 <UserCard name="Olivia" age={20} city="Haripur" />  
 </div>  
 );  
}

✅ Output:

Rehan  
Age: 21  
City: Islamabad  
  
Olivia  
Age: 20  
City: Haripur

### 🔹 Props with Functions

Props aren’t only strings or numbers—you can also pass **functions**.

function Button({ text, onClick }) {  
 return <button onClick={onClick}>{text}</button>;  
}  
  
function App() {  
 function handleClick() {  
 alert("Button clicked!");  
 }  
  
 return <Button text="Click Me" onClick={handleClick} />;  
}

✅ Output: - A button that shows an alert when clicked.

### 🔹 Props are Read-Only

* Props **cannot be modified** inside a child component.
* If you need to modify state/data, use **State** (coming in Day 4).

## 📝 Task (15–20 minutes)

👉 Create a ProfileCard component that accepts these props: - name (string) - age (number) - bio (string) - isStudent (boolean)

### Steps:

1. Create a ProfileCard.js component.
2. Accept props using destructuring.
3. Display the profile details in a nice layout.
4. If isStudent is true, show “Currently a student” otherwise show “Graduated”.
5. Import and use the ProfileCard component in your App.js file with at least **two different sets of props**.

✅ Expected Output Example:

Name: Rehan  
Age: 21  
Bio: Loves coding and cricket  
Status: Currently a student  
  
Name: Boss  
Age: 25  
Bio: Software Engineer  
Status: Graduated

⚡ After you complete this task, share your ProfileCard.js and App.js code, and I’ll review it for correctness!