🧭 React Props — Simplified Reference Guide

💡 Introduction

Perfect question 💡 — and trust me, almost every React beginner feels confused about **props** at first.
Let's completely simplify it — like we're learning it from zero.

🧠 What Are Props (Super Simple Version)

Props = Data you send from one component to another.

That's it.
Think of props as **function parameters**.

🧩 Example in Plain JavaScript

```
function greet(name) {
  console.log("Hello " + name);
}

greet("Manish");
```

✅ You pass `"Manish"` → the function uses it.

⚛️ Now, in React

```
function Child(props) {
  return <h2>Hello {props.name}</h2>;
}

<Child name="Manish" />
```

✅ You pass `"Manish"` → the component uses it.

🎨 Step-by-Step Visual Explanation

Step 1️⃣— Parent Sends Data

```
<Child name="Manish" />
```

This is like saying:
📦 “Hey React, call the `Child` function and give it `{ name: 'Manish' }`.”

Step 2️⃣— Child Receives Data

```
function Child(props) {
  return <h2>Hello {props.name}</h2>;
}
```

👉 React calls this function as:

```
Child({ name: 'Manish' })
```

So props is just an **object**:

```
props = { name: "Manish" }
```

✅ Easy Example 1 — One Prop

Parent:

```
<Child name="Manish" />
```

Child:

```
function Child(props) {
  return <h2>Hello {props.name}</h2>;
}
```

Output → Hello Manish

✅ Easy Example 2 — Multiple Props

Parent:

```
<Child name="Manish" age={20} country="Nepal" />
```

Child:

```
function Child(props) {
  return (
    <h3>
      Hello, I'm {props.name}, {props.age} years old from {props.country}.
    </h3>
  );
}
```

Output → Hello, I'm Manish, 20 years old from Nepal.

✅ Easy Example 3 — Cleaner Version (Destructuring)

You can simplify it by unpacking props directly:

```
function Child({ name, age, country }) {
  return (
    <h3>
      Hello, I'm {name}, {age} years old from {country}.
    </h3>
  );
}
```

✅ Same output — cleaner code.

✅ Easy Example 4 — Passing an Object

Parent:

```
const user = { name: "Manish", age: 20, country: "Nepal" };
<Child user={user} />
```

Child:

```
function Child({ user }) {
  return (
    <h3>
      Hello, I'm {user.name}, {user.age} years old from {user.country}.
    </h3>
  );
```

```
}
```

Output → Hello, I'm Manish, 20 years old from Nepal.

📜 Summary Table

You Write in Parent	What Child Receives	How to Access
`<Child name="Manish" />`	`{ name: "Manish" }`	`props.name`
`<Child name="Manish" age={20} />`	`{ name: "Manish", age: 20 }`	`props.name`, `props.age`
`<Child user={userData} />`	`{ user: { name:..., age:... } }`	`props.user.name`
`<Child name="Manish" />` (with destructuring)	`{ name: "Manish" }`	`{ name }` → directly use `name`

🎯 Golden Rules to Remember

- React always gives **one object (`props`)** to your component.
- You can name it **props**, **data**, or anything — but it's always an object.
- You can either:
  - 👉 Access data via **`props.something`**
  - 👉 Or destructure it using **`{ something }`**

🧩 Quick Summary

- Props = Parameters for components
- They help pass data from **parent → child**
- Always read-only
- Use destructuring for clean, readable code