Props:-

Simple Definition

Props are inputs to a component.
They allow you to send data and functions from one component to another.
Props are *read-only* — a child cannot modify them.

Why do we need props?

Props help you:

- Pass data (text, numbers, arrays, objects)
- Create reusable components
- Change what a component displays
- Trigger actions in parent components (via function props)

Step-by-step guide — Examples 1–6 (props)

I'll walk you through building a small React project that demonstrates examples **1 → 6** from the earlier message: **basic props, destructuring, multiple props, arrays/objects, functions-as-props, and default props**. We'll use Vite (fast and modern). Each example includes the code, where to put it, how to run, and a short explanation.

---

0)Prepare the project (one-time)

1. **Open a terminal.**
2. **Create a new Vite React app:**

```
npm create vite@latest react-props-demo -- --template react
cd react-props-demo
npm install
```

3. **Start the dev server:**

```
npm run dev
```

Open the URL printed in terminal (usually `http://localhost:5173`).

---

1) File structure we'll use

```
react-props-demo/
├─ index.html
├─ package.json
├─ src/
│  ├─ main.jsx
│  ├─ App.jsx
│  └─ components/
│     ├─ Welcome.jsx
│     ├─ UserCard.jsx
│     ├─ Profile.jsx
│     ├─ ActionButton.jsx
│     └─ Greeting.jsx
```

You already have `main.jsx` from Vite. Replace `App.jsx` and add the components listed.

---

2) Example 1 — Basic props (Welcome)

File: `src/components/Welcome.jsx`

```
// Welcome.jsx
export default function Welcome(props) {
  return <h2>Hello, {props.name}!</h2>;
}
```

What to do: Create the file and export the component.

Usage in App: (shown below)

Explanation: Parent passes `name="Alice"`. Child reads `props.name`. Props are read-only.

---

3) Example 2 — Destructuring props (cleaner)

File: (we'll keep using `Welcome.jsx` above, but show alternative)

```
// Welcome.jsx  (alternative using destructuring)
export default function Welcome({ name }) {
  return <h2>Hello, {name}!</h2>;
}
```

What to do: Replace `props` with destructuring `{ name }`. This is just syntactic sugar for `props.name`.

---

4) Example 3 — Multiple props (UserCard)

File: `src/components/UserCard.jsx`

```
// UserCard.jsx
export default function UserCard({ name, age, isMember }) {
  return (
    <div style={{border: "1px solid #ddd", padding: 12, borderRadius: 6}}>
      <h3>{name}</h3>
      <p>Age: {age}</p>
      <p>Status: {isMember ? "Member" : "Guest"}</p>
    </div>
  );
}
```

Explanation: Parent passes several props (`name`, `age`, `isMember`). Child uses them to render different UI.

---

5) Example 4 — Arrays & objects as props (Profile)

File: `src/components/Profile.jsx`

```
// Profile.jsx
export default function Profile({ user, hobbies }) {
  return (
    <div style={{marginTop: 12}}>
```

```
      <h2>{user.name} ({user.age})</h2>
      <ul>
        {hobbies.map((hobby, idx) => (
          <li key={idx}>{hobby}</li>
        ))}
      </ul>
    </div>
  );
}
```

Explanation: You can pass objects (`user`) and arrays (`hobbies`). Use `map()` to render lists; include `key` for list items.

---

6) Example 5 — Passing functions as props (ActionButton)

File: `src/components/ActionButton.jsx`

```
// ActionButton.jsx
export default function ActionButton({ onAction, label }) {
  return (
    <button onClick={onAction}>
      {label}
    </button>
  );
}
```

Parent (in `App.jsx`) will pass a function like `() => alert('clicked')`. This is how a child triggers behavior in the parent.

---

7) Example 6 — Default props (Greeting)

File: `src/components/Greeting.jsx`

```
// Greeting.jsx
export default function Greeting({ name = "Guest" }) {
  return <h3>Welcome, {name}</h3>;
}
```

Explanation: If `name` is not passed, `"Guest"` is used automatically because of the default parameter.

---

8) Put it all together — `src/App.jsx`

Replace `src/App.jsx` with this code to include all examples and demonstrate them:

```
// App.jsx
import Welcome from "./components/Welcome";
import UserCard from "./components/UserCard";
import Profile from "./components/Profile";
import ActionButton from "./components/ActionButton";
import Greeting from "./components/Greeting";

function App() {
  const user = { name: "Alice", age: 24 };
  const hobbies = ["reading", "coding", "music"];

  function handleAction() {
    // a parent function passed down to child
    alert("Action from parent triggered by child!");
  }

  return (
    <div style={{padding: 20, fontFamily: "system-ui, sans-serif"}}>
      <h1>React Props Demo — Examples 1-6</h1>

      {/* Example 1 & 2 */}
      <section>
        <h2>1 & 2. Basic props + Destructuring</h2>
        <Welcome name="Alice" />
        <Welcome name="Bob" />
      </section>

      {/* Example 3 */}
      <section>
        <h2>3. Multiple props</h2>
        <UserCard name="John Doe" age={28} isMember={true} />
      </section>

      {/* Example 4 */}
      <section>
        <h2>4. Arrays & Objects</h2>
        <Profile user={user} hobbies={hobbies} />
      </section>

      {/* Example 5 */}
      <section>
        <h2>5. Functions as props</h2>
        <ActionButton onAction={handleAction} label="Click me" />
      </section>

      {/* Example 6 */}
      <section>
        <h2>6. Default props</h2>
        <Greeting />         {/* uses default "Guest" */}
```

```
        <Greeting name="Sam" />
      </section>
    </div>
  );
}

export default App;
```

---

9) Ensure your entry file mounts `App` (Vite default)

`src/main.jsx` should look like this (created by Vite):

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
import './index.css' // optional

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

---

10) Run & verify

1. If the dev server is already running (`npm run dev`) the page should hot-reload after you save files.
2. Visit the app in browser. You should see:
    - "Hello, Alice!" and "Hello, Bob!"
    - A user card for John Doe
    - Profile for Alice with a list of hobbies
    - A button that triggers an alert when clicked
    - Greeting with default and provided name