ChatGPT

React Props — Explained Simply for Reference

🔍 What are Props?

- **Props** (short for "properties") are used to pass data **from parent component to child component**.
- Think of props like **arguments passed to a function**.
- They help make components reusable and dynamic.

App.jsx — Using a Component and Passing Props

In your `App.jsx` (the parent component), you use child components and pass props like this:

```jsx
import GreetUser from "./GreetUser";

function App() {
  return <GreetUser username="Raj" />;
}

export default App;
```

- `username="Raj"` is the **prop** being passed
- `GreetUser` is the **child component** that receives the prop

GreetUser.jsx — Receiving and Using Props

In the child component (e.g. `GreetUser.jsx`), you receive and use the props:

```jsx
function GreetUser(props) {
  return <p>Welcome, {props.username}!</p>;
}

export default GreetUser;
```

Or use **destructuring** for cleaner code:

```jsx
function GreetUser({ username }) {
  return <p>Welcome, {username}!</p>;
}
```

- `props` is an object automatically passed by React

- `username` is accessed either by `props.username` or via destructuring

Why Use Props?

- To **reuse components** with different data
- To **keep components pure** (no side effects or internal state)

PropTypes (for Type Checking)

Install:

```
npm install prop-types
```

Usage:

```
import PropTypes from "prop-types";

function GreetUser({ username }) {
  return <p>Hello, {username}</p>;
}

GreetUser.propTypes = {
  username: PropTypes.string.isRequired,
};
```

- Helps catch bugs by warning if incorrect type is passed

Dynamic Rendering Using Props

```
function StatusMessage({ isLoggedIn }) {
  return isLoggedIn ? <p>Welcome back!</p> : <p>Please log in.</p>;
}
```

Passing Different Data Types as Props

You can pass strings, numbers, booleans, arrays, objects, and even functions:

```
function DisplayInfo({ name, age, isAdmin }) {
  return (
    <div>
      <p>Name: {name}</p>
      <p>Age: {age}</p>
```

```
      <p>Status: {isAdmin ? "Admin" : "User"}</p>
    </div>
  );
}

<DisplayInfo name="Raj" age={25} isAdmin={true} />
```

- Strings: `name="Raj"`
- Numbers: `age={25}`
- Booleans: `isAdmin={true}` or `isAdmin={false}`

📊 Summary

- Props allow components to be **dynamic and reusable**
- They flow **one-way (parent → child)**
- You can use **destructuring** to clean up your code
- Use **PropTypes** for debugging and reliability
- You can pass **any data type** as a prop — string, number, boolean, object, function, etc.

You're crushing React basics, Raj! This `props` cheat sheet will be a great reference as your components grow!