React Props (Properties)

Props (short for **properties**) are a way to pass data from a **parent component** to a **child component** in React. They allow components to be **reusable** and **dynamic** by accepting different values.

How Props Work

1. **Passing Props** – Props are passed as attributes to a child component in JSX.
2. **Receiving Props** – The child component receives props as a function parameter.
3. **Using Props** – Props can be used within the child component to render dynamic content.

Example: Using Props

1. Passing Props from Parent to Child

```
import React from "react";
import ReactDOM from "react-dom";

const Greeting = (props) => {
  return <h1>Hello, {props.name}!</h1>;
};

const App = () => {
  return (
    <div>
      <Greeting name="Alice" />
      <Greeting name="Bob" />
    </div>
  );
};
```

```
ReactDOM.render(<App />, document.getElementById("root"));
```

Explanation:

- The App component passes the name prop to the Greeting component.
- The Greeting component receives props and uses props.name to display the name dynamically.

Destructuring Props

Instead of using props.name, we can **destructure** props directly.

```
const Greeting = ({ name }) => {
  return <h1>Hello, {name}!</h1>;
};
```

Multiple Props Example

Props can contain multiple values.

```
const UserProfile = ({ name, age }) => {
  return (
    <div>
      <h2>Name: {name}</h2>
      <p>Age: {age}</p>
    </div>
  );
};

const App = () => {
  return <UserProfile name="John Doe" age={30} />;
};
```

Default Props

If a prop is not provided, we can set a default value using defaultProps.

```
const Greeting = ({ name = "Guest" }) => {
  return <h1>Hello, {name}!</h1>;
};
```

OR

```
Greeting.defaultProps = {
  name: "Guest",
};
```

Props are Read-Only

Props **cannot** be modified inside a component.

```
const Greeting = (props) => {
  props.name = "John"; // ❌ This will cause an error
  return <h1>Hello, {props.name}!</h1>;
};
```

Instead, props should remain **immutable**, and data should be updated in the **parent component**.

Passing Functions as Props

Props can also be functions, allowing communication from **child to parent**.

```
const Button = ({ handleClick }) => {
  return <button onClick={handleClick}>Click Me</button>;
};

const App = () => {
  const showAlert = () => alert("Button clicked!");
  return <Button handleClick={showAlert} />;
};
```

Conclusion

- Props **pass data** from parent to child.
- They are **immutable** (read-only).
- Props can be **strings, numbers, objects, functions**, etc.
- Use **defaultProps** to set default values.
- Destructuring makes code **cleaner**.