WEB APP DEVELOPMENT WITH REACTJS (INT252)

Lecture 7: Functional Components & Props

Unit II – Component Design & Styling

Syllabus Mapping:

- Functional components
- Props (data passing)
- Component reusability

Target Learner: Beginner (HTML, CSS, JavaScript knowledge)

Why Components Matter

React apps are NOT written as one big file.

They are built using:

- Small
- Reusable
- Independent components

What is a Component?

A component is:

- A JavaScript function
- That returns JSX
- Represents part of UI

Component Analogy

Think of components like:

```
LEGO Blocks 🧱
```

Small pieces combined to build big apps

Functional Components (Modern React)

React uses **functions**, not classes.

```
function Welcome() {
  return <h1>Welcome</h1>;
}
```

Component Naming Rule

❗ Component names MUST:

- Start with **capital letter**

```
function header() {} // ❌
function Header() {} // ✅
```

Why Capital Letters Matter

React treats:

- lowercase → HTML tag
- uppercase → Component

Using Components Inside Components

```
function Header() {
  return <h2>Header</h2>;
}

function App() {
  return <Header />;
}
```

Components Are Reusable

```
<Header />
<Header />
<Header />
```

Same component, multiple times

Problem: Static Components

Hardcoded data ❌

```
<h1>Hello Aman</h1>
```

Not reusable

Solution: Props

Props = Properties

Used to:

- Pass data to components

Props Example (Basic)

```
function Welcome(props) {
  return <h1>Hello {props.name}</h1>;
}
```

Passing Props

```
<Welcome name="Aman" />
<Welcome name="Student" />
```

Props Flow (Diagram)

```
Parent Component
      ↓ props
Child Component
```

Props Are Read-Only

❌ This is NOT allowed:

```
props.name = "New";
```

Props cannot be modified

Destructuring Props (Recommended)

```
function Welcome({ name }) {
  return <h1>Hello {name}</h1>;
}
```

Cleaner syntax

Multiple Props Example

```
function User({ name, age }) {
  return <p>{name} is {age} years old</p>;
}
```

Passing Numbers & Expressions

```jsx
<User name="Aman" age={20} />
```

Strings → quotes

JS values → `{}`

Default Props Concept

```
function Greeting({ name = "User" }) {
  return <h1>Hello {name}</h1>;
}
```

Component Composition

Components can contain components:

```
App
 ├── Header
 ├── Content
 └── Footer
```

Common Beginner Mistakes

❌ Forgetting capital letter
❌ Trying to change props
❌ Passing string instead of number

Practice Exercises

1. Create a component called `Profile`
2. Pass name and course as props
3. Render it twice with different data

✅ Answers – Practice Exercises

```
function Profile({ name, course }) {
  return <p>{name} studies {course}</p>;
}

function App() {
  return (
    <>
      <Profile name="Aman" course="React" />
      <Profile name="Alex" course="JavaScript" />
    </>
  );
}
```

Key Takeaways

- Components are reusable functions
- Props pass data from parent to child
- Props are read-only
- Destructuring makes code cleaner

Next Lecture

Lecture 8: Component Composition & Conditional Rendering

Unit II – Component Design & Styling