WEB TECHNOLOGIES

React Components and JSX

Prof. Pavan A C

Department of Computer Science and Engineering

Acknowledgement:

Teaching Assistants(Harini B and Chandana MS)

React Component and JSX (Functional Comps)

What are Components

In React, components are reusable, independent code blocks (A function or a class) that define the structure and behavior of the UI. They accept inputs (props or properties) and return elements that describe what should appear on the screen.

Key Concepts of React Components:

- Each component handles its own logic and UI rendering.
- Components can be reused throughout the app for consistency.
- Components accept inputs via props and manage dynamic data using state.
- Only the changed component re-renders, not the entire page.
- Stateless by default (but can use hooks for state)
- Modern standard for React development

React Component and JSX (Functional Comps)

What are Components

Need for Components

- There may be elements which are similar
- A need to make a change in one will reflect changes in multiple places
- Similar to functions, if we could write code related to the element in one place,

changes can be minimized

- Solution - Reusable piece of JavaScript code that output (via JSX) HTML elements

React Component and JSX (Functional Comps)

Types of Components

- **Stateless Functional Component**
  - Includes simple JavaScript functions and immutable properties, i.e., the value for properties cannot be changed.
  - Use hooks to achieve functionality for making changes in properties using JS.
  - Used mainly for UI.

- **Stateful Class Component**
  - Classes which extend the Component class from React library.
  - The class component must include the render method which returns HTML.

React Component and JSX (Functional Comps)

Functional Components

- The simplest way to define a component is to write a JavaScript function:
- A functional component is simply a JavaScript function that returns JSX (JavaScript XML, the syntax that looks like HTML inside JavaScript).
- They are the building blocks of modern React apps.

```
function Welcome() {
  return <h1>Hello, World!</h1>;
}
```

Function Declaration or Arrow Functions can be used.

JSX returned must be single root element.

```
// Arrow function syntax
const Welcome = () => {
  return <h1>Hello, World!</h1>;
}
```

React Component and JSX (Functional Comps)

Functional Components

Properties are ways in which React components can be customized.

- Props are arguments passed into React components and are passed via HTML attributes.
- Properties are immutable and are same as what attributes are to HTML elements.
- Their most basic use is in the form of attributes, in JSX.
- Are actually passed to a constructor and can be accessed by this.props
- Design decision is to use the Attributes specified in hyphen-case in HTML as camelCase in JSX.
- Make the function of your component read the props from the props parameter
- When rendering the component, add the prop to the component using the attribute

React Component and JSX (Functional Comps)

Functional Components

- Props (short for properties) are how you pass data into a component.
- props is an object that contains all the values passed to the component.

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
// Usage
<Greeting name="Alice" />
```

This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element. We call such components “function components” because they are literally JavaScript functions.

Destructuring Props

Instead of using props.attributename, you can destructure the props object directly in the function's argument to make the code cleaner.

```
function UserCard({ name, email, age }) {
  return (
    <div>
      <h2>{name}</h2>
      <p>Email: {email}</p>
      <p>Age: {age}</p>
    </div>
  );
}
```

The UserCard component receives the name,email,age prop using destructuring in the function argument, making it easier to access the name value.
Instead of writing props.name, we now directly use { name } within the component.

Functional Components

- Functional components are simple, reusable JavaScript functions that return JSX and accept props to customize their output.
- They are stateless by default but form the modern foundation of React apps,enabling modular UI and code reuse.
- In larger, interactive apps, components need to manage dynamic data, handle user actions, and perform side effects (like fetching data).

Introducing React Hooks

- React Hooks empower functional components to do more—use local state, respond to lifecycle events, and share logic easily.
- Hooks such as useState (for state) and useEffect (for side effects) unlock features that were previously only possible in class components.
- **Upcoming unit** will dive deep into hooks, and how they make functional components even more powerful for modern web development.

React Component and JSX (Functional Comps)

Functional Components - Example

```
function Student() {
  const name: string = "Pavan";
  const course: string = "Web Technologies";
  const semester: number = 3;

  return (
    <div>
      <h2>Student Information</h2>
      <p>Name: {name}</p>
      <p>Course: {course}</p>
      <p>Semester: {semester}</p>
    </div>
  );
}
```

```
// Render Student component
ReactDOM.createRoot(document.getElementById("root1")).render(

    <Student />

);
```

React Component and JSX (Functional Comps)

Functional Components - Example

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

// Rendering the component
ReactDOM.createRoot(document.getElementById("root")).render(
  <Welcome name="Pavan" />
);
```

React Component and JSX (Functional Comps)

Functional Components - Example

```
function App() {
  function handleClick() {
    alert("Button was clicked!");
  }

  return (
    <div>
      <h1>Hello, JSX!</h1>
      <p>2 + 2 = {2 + 2}</p>
      <button onClick={handleClick}>Click Me</button>
    </div>
  );
}
```

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

THANK YOU

Prof. Pavan A C

Department of Computer Science and Engineering