**Hands-on: 7. ReactJS-HOL**

**React Component Data Management: Props, State, and Rendering**

React's core strength lies in its ability to build dynamic and reusable UI components. A fundamental aspect of this is understanding how to pass data between components using props and manage dynamic data within components using state. Grasping props, default props, state, and ReactDOM.render() is crucial for mastering React.

**Defining Props**

Props (short for properties) are read-only attributes used to pass data from a parent component to a child component in React.

**Key Points about Props:**

* Props are passed as HTML-like attributes.
* Props are immutable inside the child component, meaning they cannot be changed by the child.
* Props enable component reusability with different values.

**Example:**

JavaScript

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

Usage: <Greeting name="Tanishq" />

**Explaining Default Props**

Default Props provide default values for props when a parent component does not explicitly pass them.

**Purpose:**

* To prevent undefined values when props are missing.
* To ensure the component renders with fallback data.

**Example:**

JavaScript

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

Greeting.defaultProps = {

name: "Guest"

};

If 'name' is not passed, it will default to "Guest".

**Differences Between State and Props**

| Aspect | Props | State |
| --- | --- | --- |
| Definition | External data passed to a component | Internal data managed within the component |
| Mutability | Immutable (read-only) | Mutable (can be changed using setState) |
| Ownership | Set by the parent component | Managed by the component itself |
| Purpose | Configures component behavior | Tracks dynamic UI or logic changes |
| Usage | Used in both functional and class components | Used in both, but often initialized in class or with useState in functions |

**Explaining ReactDOM.render()**

The ReactDOM.render() method is used to render React elements or components into the actual Document Object Model (DOM).

Syntax:

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

**How It Works:**

* It takes a React element or component as its first argument.
* It takes a DOM node (typically a div with id='root') as its second argument.
* It renders the React virtual DOM into the actual browser DOM.

Note for Modern React (v18+):

In modern React versions (v18+), ReactDOM.render() has been replaced by createRoot for improved performance and new features.

**Modern Usage Example:**

JavaScript

import { createRoot } from 'react-dom/client';

const root = createRoot(document.getElementById('root'));

root.render(<App />);

Understanding props, default props, state, and ReactDOM.render() (or createRoot) is foundational for mastering React's component-driven architecture. While props facilitate component customization and reuse, state enables components to be dynamic. The ReactDOM.render() (or createRoot) method acts as the entry point for rendering the entire application into the DOM. Mastery of these concepts is essential for building responsive and efficient React applications.