**3. ReactJS-HOL**

**• Explain React components**

React components are the building blocks of a React application. A component is a reusable piece of UI that can have its own logic and data. Components can be nested inside other components, enabling complex interfaces to be built from simple units.

* Think of components like functions that return HTML elements.
* They can manage their own state and props (input data).

**• Identify the differences between components and JavaScript functions**

| **Aspect** | **React Component** | **JavaScript Function** |
| --- | --- | --- |
| **Purpose** | Returns UI (JSX), used in rendering views | Performs logic or computation |
| **Syntax** | Uses JSX and may include React-specific features | Pure JavaScript syntax |
| **Lifecycle** | Has lifecycle methods (in class components) | No lifecycle or React state management |
| **State Management** | Can manage state (useState, this.state) | No built-in state |
| **Reusability** | Designed for UI reuse | Meant for logic reuse |

**• Identify the types of components**

React has two main types of components:

1. Class Components
   * Use ES6 classes.
   * Can have lifecycle methods.
   * Manage state using this.state.
2. Function Components
   * Use JavaScript functions.
   * Simpler and easier to write.
   * Use React Hooks (like useState, useEffect) to manage state and lifecycle.

**• Explain class component**

A class component is a React component defined using an ES6 class that extends React.Component. It has access to state, lifecycle methods, and the render() method which returns JSX.

Example:

jsx

import React, { Component } from 'react';

class Welcome extends Component {

render() {

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

}

}

**• Explain function component**

A function component is a simpler way to write React components using plain JavaScript functions. They accept props and return JSX. Since React 16.8, function components can also use hooks for managing state and side effects.

Example:

jsx

function Welcome(props) {

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

}

Using hooks:

jsx

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return <button onClick={() => setCount(count + 1)}>Click {count}</button>;

}

**• Define component constructor**

The constructor() is a special method in class components used to:

* Initialize state.
* Bind event handlers.
* Set up any initial configuration.

Syntax:

jsx

constructor(props) {

super(props);

this.state = { count: 0 };

}

Note: You must call super(props) before using this.

**• Define render() function**

The render() method is required in class components. It returns the JSX (HTML-like syntax) that should be displayed on the screen.

Key Points:

* Called automatically when the component is rendered or updated.
* Must return a single parent element.

Example:

jsx

render() {

return <div>Hello, World!</div>;

}