### ****Objective Answers – Week6\_HandsOn3****

#### 1. Explain React components

React components are the core building blocks of any React application. Each component represents a self-contained, reusable section of the user interface. Components in React can be nested, managed, and reused across the app, which helps in developing complex user interfaces efficiently. They encapsulate their own logic and rendering, making code modular and maintainable. For example, a component could represent a user profile card, a navigation bar, or a form input field — each responsible for its own structure and behavior.

#### 2. Identify the differences between components and JavaScript functions

While React components and JavaScript functions may look similar, they serve different purposes:

* **Purpose**: A JavaScript function performs a task or returns a value, whereas a React component returns JSX to render UI.
* **Usage**: React components follow specific conventions (capitalized names, return JSX, used in a render tree). JS functions are general-purpose.
* **Lifecycle & State**: React components can use state, hooks, and lifecycle methods (in class components). Regular JS functions have no such capability.
* **Reactivity**: React components automatically re-render when props/state change. JavaScript functions do not track changes.

#### 3. Identify the types of components

React provides two main types of components:

* **Class Components**: These are ES6 classes that extend React.Component. They have access to state, props, and lifecycle methods like componentDidMount() and componentWillUnmount(). Class components are ideal when complex logic or state management is required.
* **Function Components**: Introduced as stateless components, they’ve evolved (with hooks) to handle state and side effects. They are simpler, more concise, and are now the preferred standard for writing React components in modern development.

#### 4. Explain class component

A **class component** in React is defined using the ES6 class syntax and must extend React.Component. It includes a mandatory render() method that returns JSX, which defines what will be displayed in the UI. Class components can hold and manage internal state using this.state and can utilize lifecycle methods to run code at specific points in the component's life (mounting, updating, unmounting). While still supported, class components are being gradually replaced by function components with hooks in most modern projects.

Example:

class Welcome extends React.Component {

render() {

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

}

}

#### 5. Explain function component

A **function component** is a simple JavaScript function that accepts props as input and returns JSX as output. These are also called **stateless components**, but with the introduction of hooks like useState and useEffect, function components can now handle state and side effects, making them as powerful as class components. Their simplicity and better performance have made them the new standard in React.

Example:

function Welcome(props) {

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

}

#### 6. Define component constructor

In class components, the **constructor** is a special method used to initialize the component's state and bind event handler methods. It is the first method called when a component is created. When defining a constructor in a React class component, you must call super(props) before accessing this, as it initializes the parent React.Component class.

Example:

constructor(props) {

super(props);

this.state = { count: 0 };

}

You usually use the constructor to set initial values for the state or bind custom methods to this.

#### 7. Define render() function

The render() function is required in every class component. It returns the JSX that should be rendered to the screen. When the component’s state or props change, the render() method is called again to update the UI. It must return a single enclosing element (like a <div> or a React fragment <>...</>).

Example:

render() {

return (

<div>

<h1>Welcome to React</h1>

</div>

);

}

The render() method should remain **pure**, meaning it should only return JSX and not cause any side effects like API calls or modifying state directly.