### ****Objective Answers – Week6\_HandsOn2****

#### 1. Explain React components

React components are the building blocks of any React application. They represent individual, reusable pieces of the user interface (UI). Each component encapsulates its own structure, logic, and styling, making the code more modular and easier to maintain. Components can be thought of as custom HTML elements that React interprets and renders on the web page. React automatically updates the components whenever the data (props or state) changes, ensuring the UI stays in sync with the application logic. Components can either be **class-based** or **function-based**, depending on how they are defined.

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

While both React components and JavaScript functions can receive inputs (arguments or props) and return outputs (JSX or values), their purposes and behavior differ:

* A regular **JavaScript function** is used for general computation and logic, whereas a **React component** is specifically designed to return JSX and render UI elements.
* React components follow certain conventions like starting with a capital letter and returning JSX.
* Components can maintain their own internal **state**, whereas plain JavaScript functions do not hold persistent UI state unless they’re enhanced using hooks (like useState).
* React components are re-rendered automatically when their props or state change, while JavaScript functions execute only when called.

#### 3. Identify the types of components

React primarily supports two types of components:

* **Class Components**: These are ES6 classes that extend the React.Component class. They can have lifecycle methods (like componentDidMount) and their own internal state using this.state.
* **Functional Components**: These are simpler components defined as JavaScript functions. Originally stateless, they can now handle state and lifecycle behavior using **React Hooks** like useState and useEffect.

Functional components are the modern standard due to their simplicity and performance. However, class components are still widely used in legacy codebases.

#### 4. Explain class component

A **class component** in React is defined using ES6 class syntax. It must extend the base class React.Component and should implement a render() method, which returns the JSX to be displayed in the UI.

Class components can:

* Hold and manage local state using this.state
* Access lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount
* Be passed props and return JSX conditionally or dynamically based on logic

Example:

class MyComponent extends React.Component {

render() {

return <h1>Hello from class component</h1>;

}

}

#### 5. Explain function component

A **function component** is a simple JavaScript function that returns JSX (React’s syntax extension for HTML in JavaScript). It accepts props as an argument and renders UI based on those props. Initially, functional components were stateless, but with the introduction of **React Hooks**, they can now maintain state, side effects, context, and more.

Example:

function MyComponent(props) {

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

}

Functional components are preferred in modern React development due to their concise syntax, better performance, and compatibility with hooks.

#### 6. Define component constructor

In class components, the **constructor** is a special method that is automatically called when an instance of the component is created. It is used to initialize the component’s **state** and bind methods.

The constructor must call super(props) before using this, as it calls the parent class constructor (React.Component).

Example:

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

}

The constructor is optional unless you need to use or initialize state or bind methods to this.

#### 7. Define render() function

The render() function is a **mandatory** method in a class component. It defines **what the UI should look like** by returning JSX. Whenever the component’s state or props change, the render() method is called again to reflect those updates in the DOM.

Example:

class MyComponent extends React.Component {

render() {

return <div>Hello World</div>;

}

}

This method must return only one parent element (a div, section, or React Fragment) that wraps the entire JSX structure.