**Cognizant Digital Nurture 4.0**

**Week-6**

**Name -Brishti Das**

**Superset id. -6363729**

**React Handson Exercise 2**

### Explain React Components

**React components** are the building blocks of any React application. They are reusable, self-contained pieces of UI that can manage their own state and be composed to build complex user interfaces.

A component returns **JSX**, which describes what the UI should look like.

They can be rendered independently and reused across the app.

Components can be either **class-based** or **function-based**.

### Identify the Differences Between Components and JavaScript Functions

| **Aspect** | **React Component** | **JavaScript Function** |
| --- | --- | --- |
| **Purpose** | Used to build UI elements | Performs a specific task or calculation |
| **Return Type** | Returns JSX (UI structure) | Returns any JavaScript data (e.g., number, string) |
| **Integration with React** | Integrated into React rendering cycle | Not connected with React lifecycle |
| **Can hold state** | Yes (especially class components) | No (unless managed with external tools) |
| **Lifecycle methods** | Available (class components) | Not applicable |

### Identify the Types of Components

React has two primary types of components:

**Function Components**

**Class Components**

### Explain Class Component

A **class component** is a more traditional way to create a component in React. It uses JavaScript ES6 class syntax and can manage its own state and lifecycle methods.

**Syntax:**

jsx

CopyEdit

import React, { Component } from 'react';

class Welcome extends Component {

render() {

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

}

}

**Key Features:**

Inherits from React.Component

Has access to this.props and this.state

Lifecycle methods like componentDidMount, componentDidUpdate can be used.

### Explain Function Component

A **function component** is a simpler way to write components using JavaScript functions. It was originally stateless, but with React Hooks, it can now manage state and side-effects.

**Syntax:**

jsx

CopyEdit

function Welcome(props) {

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

}

**With Hooks (for state):**

jsx

CopyEdit

import React, { useState } from 'react';

function Counter() {

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

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

}

### Define Component Constructor

In **class components**, the constructor is a special method used to initialize state and bind methods.

**Example:**

jsx

CopyEdit

class Welcome extends React.Component {

constructor(props) {

super(props);

this.state = { name: 'Guest' };

}

render() {

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

}

}

super(props) must be called before using this

Used to set initial state and bind event handlers

### Define render() Function

The render() method is a **mandatory method in class components**. It returns the JSX that defines the component’s UI.

**Example:**

jsx

CopyEdit

class MyComponent extends React.Component {

render() {

return <div>Hello World</div>;

}

}

It is called whenever the component needs to display or re-display.

It must return a single JSX element or null.