**WEEK-6: 2. REACT JS – HOL**

**Name:** Abhignya Gitti

**Superset ID:** 6374632

* **Explain React components**

React components are the building blocks of a react application. These are reusable and independent blocks of code that are used to define and manage the UI logic. They resemble JavaScript functions which take an input (props) and return an output (JSX).

There are two types of components: Class components and Functional components.

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

|  |  |
| --- | --- |
| React Components | Plain JavaScript Functions |
| Must be capitalized like “MyComponent” and used in JSX like <MyComponenets/> | Can have any name casing like “mycomponent” |
| Expected to return JSX | May return JSX but they are used directly or mostly invoked in code |
| Can maintain state(class components) or use hooks(functional components) | Does not support lifecycle or state |
| function getGreeting(name) {  return <div>Hello, {name}!</div>;  }  function MyComponent(props) {  return <div>{getGreeting(props.name)}</div>;  } | function MyComponent(props) {  return <div>Hello, {props.name}!</div>;  }  // Usage in JSX:  <MyComponent name="Sara" /> |

* **Identify the types of components**

React Components can be classified into two types:

* Functional Components:

These are JS Functions that accept “Props” (properties) as an argument and return JSX elements.

* Class Components: These are ES6 classes that extend React Component.
* **Explain class component**

**Class Components** are React components defined using ES6 classes that extend from React.Component (or its base class). Prior to Hooks (React 16.8), class components were the primary way to manage internal state and lifecycle methods in React applications.

**Ex:**

import React from 'react';

class Greeting extends React.Component {

render() {

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

}

}

* **Explain function component**

A **React functional component** is just a plain JavaScript function (its name starts with a capital letter) that takes a special input called props and returns **JSX**, which is a mix of HTML-like tags inside JavaScript. It’s the easiest and cleanest way to make UI in React. Since the new version of React, you can also use **Hooks** like useState and useEffect inside these functions to manage state and side-effects—just like in class components, but with much less code and no “this” keyword.

Ex:

import React, { useState } from 'react';

function Greeting(props) {

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

return (

<div>

<p>Hello, {props.name}! You clicked {count} times.</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}

* **Define component constructor**

In a **React class component,** the constructor(props) method is a special function that runs automatically when a component is created. It’s used to set up the component before it's shown on the screen. Mostly, it’s used to **initialize the component’s state** (this.state = {...}). You must call super(props) at the start to let React set up internal behavior properly. If you don’t need state or binding, you can skip defining a constructor entirely—it’s optional.

* **Runs first, before render**: Always executes when the component is created.
* **Call** super(props): Required before using this.props inside the constructor.
* **Define render() function**

In a React **class component**, the render () method is the only required function. It defines what your component should display on screen by returning **JSX** (JavaScript + HTML-like code). React calls render () automatically whenever the component's **props** or **state** change. React uses the output from render() to build a **virtual DOM tree**, which it then compares to the previous version to find the minimal updates needed in the real DOM. This makes the UI update efficiently and only when necessary.