**1. Explain React Components**

**React components** are reusable, independent pieces of UI in a React application. Each component is responsible for rendering part of the user interface and can manage its own logic and state.

* Components let you split the UI into smaller, manageable parts.
* They receive **props** (input) and can maintain **state** (internal data).
* Components can be reused across different parts of the app.

**2. Identify the Differences Between Components and JavaScript Functions**

| **Aspect** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| **Purpose** | Used to build UI elements | Used for logic, calculations, or data tasks |
| **Return Value** | Returns JSX (UI) | Returns any data (number, string, etc.) |
| **State Management** | Can manage state (especially with hooks) | Cannot manage state |
| **Lifecycle Hooks** | Can use lifecycle methods or hooks | No lifecycle support |
| **Integration with React** | Built to work with React rendering | Not integrated with React's rendering |

**3. Identify the Types of Components**

React has two main types of components:

1. **Class Components**
   * Defined using ES6 class syntax
   * Can use lifecycle methods and this.state
2. **Function Components**
   * Defined using JavaScript functions
   * Use **Hooks** like useState, useEffect for state and logic

**4. Explain Class Component**

A **class component** is a component written using the class keyword. It must extend React.Component and define a render() method that returns JSX.

**Example:**

import React, { Component } from 'react';

class Greeting extends Component {

render() {

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

}

}

**Features:**

* Can hold local **state**
* Uses **this.props** and **this.state**
* Supports **lifecycle methods**

**5. Explain Function Component**

A **function component** is a simpler component written as a plain JavaScript function. With React Hooks, function components can now manage state and side effects.

**Example:**

function Greeting(props) {

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

}

With Hooks:

import React, { useState } from 'react';

function Counter() {

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

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

}

**Features:**

* Simpler syntax
* Uses **Hooks** like useState, useEffect
* No need for this keyword

**6. Define Component Constructor**

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

**Syntax:**

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

}

**Key Points:**

* Must call super(props) to access this.props
* Used to set initial state and bind methods

**7. Define render() Function**

The **render()** function is a required method in class components. It defines what the UI should look like.

**Example:**

class MyComponent extends React.Component {

render() {

return <h1>Hello World</h1>;

}

}

**Key Points:**

* Returns **JSX**
* Called automatically by React during the UI update process
* Must return only **one parent element**