React-3

**1. Explain React Components**

React components are reusable, independent building blocks in a React application. Each component encapsulates its own logic and UI structure and is responsible for rendering part of the user interface using JSX.

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

| **Aspect** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| Purpose | Used for creating and managing UI | Used for performing specific tasks or calculations |
| Syntax | Includes JSX and React-specific features | Pure JavaScript syntax without JSX |
| Return Value | Returns JSX (UI elements) | Returns primitive data or objects |
| Usage | Part of React render tree | General-purpose programming logic |
| Lifecycle & State | Can use React state and lifecycle hooks | No built-in state or lifecycle management |

**3. Identify the Types of Components**

There are two primary types of React components:

* **Class Components:** Built using ES6 classes and capable of maintaining internal state and lifecycle methods.
* **Function Components:** Defined using functions. With React Hooks, they can manage state and side effects effectively.

**4. Explain Class Component**

A class component in React is created using ES6 class syntax and extends React.Component. It includes a render() method that returns JSX and can maintain state and respond to lifecycle events.

**5. Explain Function Component**

A function component is a simpler form of a component defined as a JavaScript function. It returns JSX and, with React Hooks, can manage state and side effects.

Example:

function Greet() {

return <h1>Hello Student</h1>;

}

**6. Define Component Constructor**

The constructor is a special method in a class component that is used to initialize state and bind class methods. It is called before the component is mounted.

Example:

constructor(props) {

super(props);

this.state = { score: 0 };

}

**7. Define render() Function**

The render() function is mandatory in class components. It returns the JSX to be displayed in the UI. React automatically invokes this function to render or re-render the component.

**Hands-on Lab Instructions Summary:**

To build and run the “Score Calculator” React application:

1. **Create a React application named scorecalculatorapp:**
2. npx create-react-app scorecalculatorapp
3. **Inside the src folder, create a folder named Components. Add a file named CalculateScore.js.**
4. **In CalculateScore.js, define the function component as:**
5. import React from 'react';
6. import '../Stylesheets/mystyle.css';
7. function CalculateScore(props) {
8. const { name, school, total, goal } = props;
9. const average = total / goal;
10. return (
11. <div className="score-box">
12. <h2>Score Calculator</h2>
13. <p>Name: {name}</p>
14. <p>School: {school}</p>
15. <p>Total Marks: {total}</p>
16. <p>Goal: {goal}</p>
17. <p>Average Score: {average.toFixed(2)}</p>
18. </div>
19. );
20. }
21. export default CalculateScore;
22. **Create a folder named Stylesheets and add a CSS file named mystyle.css:**
23. .score-box {
24. background-color: #f0f8ff;
25. border: 1px solid #ccc;
26. padding: 20px;
27. border-radius: 10px;
28. width: 300px;
29. margin: 20px auto;
30. text-align: left;
31. }
32. **Edit App.js to render the CalculateScore component with sample data:**
33. import React from 'react';
34. import CalculateScore from './Components/CalculateScore';
35. function App() {
36. return (
37. <div className="App">
38. <CalculateScore name="John Doe" school="ABC School" total={450} goal={5} />
39. </div>
40. );
41. }
42. export default App;
43. **Start the React application:**
44. npm start
45. **Open your browser and go to:** http://localhost:3000