## **Objectives**

* Explain React components

React components are reusable, independent pieces of UI that define how parts of your application appear and behave. They can be thought of as JavaScript functions or classes that return JSX (React’s syntax extension that looks like HTML) to render UI on the screen.

* Identify the differences between components and JavaScript functions

|  |  |  |
| --- | --- | --- |
| Aspect | React Components | Javascript Functions |
| Purpose | Produce UI elements using JSX | Perform general computations or logic |
| Return value | JSX(which React renders to the DOM) | Any Javascript value |
| React Features | Used within React applications to build UI | Used anywhere in Javascript code |

* Identify the types of components

React has two main types of components:

1. **Class Components:** Created using ES6 classes. They can hold state and use lifecycle methods.
2. **Function Components:** Created using JavaScript functions. With React Hooks, they can now also manage state and side effects.

* Explain class component

A class component is a React component defined as a JavaScript class, extending React Component. It must define a render() method that returns JSX. It can maintain internal state and use lifecycle methods.

* Explain function component

A function component is a React component defined as a JavaScript function that returns JSX. They are simpler, and with React Hooks, they can use state and side effects.

* Define component constructor

The **constructor** is a special method in class components used for initializing state and binding methods. It runs once when the component is created.

* Define render() function

The render() function is required in class components. It describes what the UI should look like by returning JSX. It is called whenever the component’s state or props change

In this hands-on lab, you will learn how to:

* Create a function component
* Apply style to components
* Render a component

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

1. Create a React project named “scorecalculatorapp” type the following command in terminal of Visual studio:

npx create-react-app scorecalculatorapp;

1. Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”
2. Type the following code in CalculateScore.js
3. import React, { Component } from 'react';
4. import './StyleSheets/CalculateScore.css';
5. class CalculateScore extends Component {
6. constructor(props) {
7. super(props);
8. this.state = {
9. score1: '',
10. score2: '',
11. score3: '',
12. total: 0
13. };
14. }
15. handleChange = (e) => {
16. this.setState({ [e.target.name]: e.target.value });
17. }
18. calculateTotal = () => {
19. const { score1, score2, score3 } = this.state;
20. const num1 = parseInt(score1) || 0;
21. const num2 = parseInt(score2) || 0;
22. const num3 = parseInt(score3) || 0;
23. this.setState({ total: num1 + num2 + num3 });
24. }
25. render() {
26. return (
27. <div className="score-container">
28. <h2>Score Calculator</h2>
29. <label>
30. Score 1:
31. <input
32. type="number"
33. name="score1"
34. value={this.state.score1}
35. onChange={this.handleChange}
36. className="score-input"
37. />
38. </label>
39. <br />
40. <label>
41. Score 2:
42. <input
43. type="number"
44. name="score2"
45. value={this.state.score2}
46. onChange={this.handleChange}
47. className="score-input"
48. />
49. </label>
50. <br />
51. <label>
52. Score 3:
53. <input
54. type="number"
55. name="score3"
56. value={this.state.score3}
57. onChange={this.handleChange}
58. className="score-input"
59. />
60. </label>
61. <br />
62. <button className="score-btn" onClick={this.calculateTotal}>Calculate Total</button>
63. <h3>Total Score: <span className="total-value">{this.state.total}</span></h3>
64. </div>
65. );
66. }
67. }
68. export default CalculateScore;
69. Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:
70. .score-container {
71. background: #f8f9fa;
72. border: 1px solid #ccc;
73. border-radius: 8px;
74. max-width: 350px;
75. margin: 30px auto;
76. padding: 30px 20px;
77. box-shadow: 0 2px 8px rgba(60,60,60,0.07);
78. text-align: center;
79. }
80. .score-input {
81. margin-left: 10px;
82. margin-bottom: 10px;
83. padding: 5px 10px;
84. width: 80px;
85. border: 1px solid #bbb;
86. border-radius: 4px;
87. font-size: 1rem;
88. }
89. .score-btn {
90. margin: 20px 0;
91. background: #2979ff;
92. color: #fff;
93. border: none;
94. padding: 8px 22px;
95. border-radius: 5px;
96. font-size: 1.08rem;
97. cursor: pointer;
98. transition: background 0.2s;
99. }
100. .score-btn:hover {
101. background: #0056b3;
102. }
103. .total-value {
104. color: #1b5e20;
105. font-weight: bold;
106. }
107. Edit the App.js to invoke the CalculateScore functional component as follows:

import React from 'react';

import CalculateScore from './CalculateScore';

function App() {

  return (

    <div>

      <CalculateScore />

    </div>

  );

}

export default App;

1. In command Prompt, navigate into scorecalculatorapp and execute the code by typing the following command:

npm start

1. Open browser and type “localhost:3000” in the address bar:

