|  |  |
| --- | --- |
| **ch26.ReactJS CSS** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# React CSS

CSS in React is used to style the React App or Component. The **style** attribute is the most used attribute for styling in React applications, which adds dynamically-computed styles at render time. It accepts a JavaScript object in **camelCased** properties rather than a CSS string. There are many ways available to add styling to your React App or Component with CSS. Here, we are going to discuss mainly **four** ways to style React Components, which are given below:

1. Inline Styling
2. CSS Stylesheet
3. CSS Module
4. Styled Components

## **1. Inline Styling**

The inline styles are specified with a JavaScript object in camelCase version of the style name. Its value is the style?s value, which we usually take in a string.

### **Example**

**App.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
4. **class** App **extends** React.Component {
5. render() {
6. **return** (
7. <div>
8. <h1 style={{color: "Green"}}>Hello JavaTpoint!</h1>
9. <p>Here, you can find all CS tutorials.</p>
10. </div>
11. );
12. }
13. }
14. export **default** App;

#### Note:**You can see in the above example, we have used two curly braces in:** <h1 style={{color: "Green"}}>Hello JavaTpoint!</h1>**. It is because, in JSX, JavaScript expressions are written inside curly braces, and JavaScript objects also use curly braces, so the above styling is written inside two sets of curly braces {{}}.**

**Output**



### **camelCase Property Name**

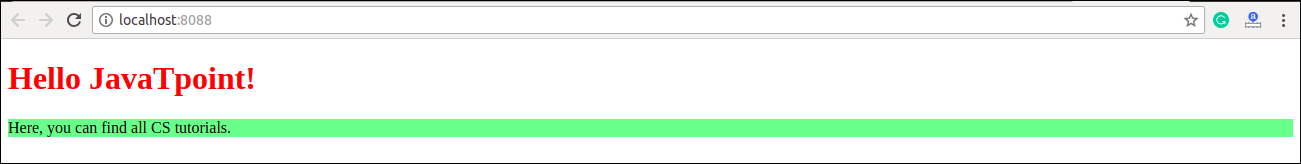
If the properties have two names, like **background-color**, it must be written in camel case syntax.

**Example**

**App.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
4. **class** App **extends** React.Component {
5. render() {
6. **return** (
7. <div>
8. <h1 style={{color: "Red"}}>Hello JavaTpoint!</h1>
9. <p style={{backgroundColor: "lightgreen"}}>Here, you can find all CS tutorials.</p>
10. </div>
11. );
12. }
13. }
14. export **default** App;

**Output**



### **Using JavaScript Object**

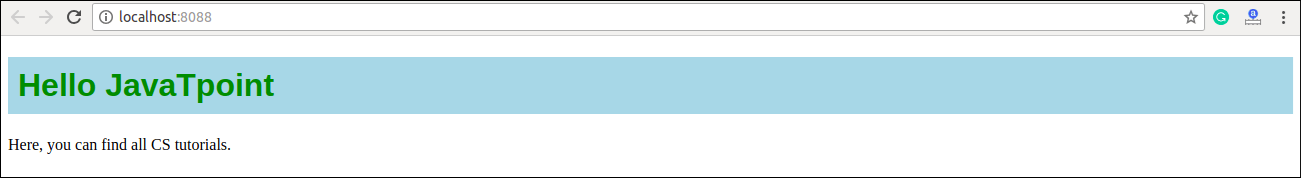
The inline styling also allows us to create an object with styling information and refer it in the style attribute.

**Example**

**App.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
4. **class** App **extends** React.Component {
5. render() {
6. **const** mystyle = {
7. color: "Green",
8. backgroundColor: "lightBlue",
9. padding: "10px",
10. fontFamily: "Arial"
11. };
12. **return** (
13. <div>
14. <h1 style={mystyle}>Hello JavaTpoint</h1>
15. <p>Here, you can find all CS tutorials.</p>
16. </div>
17. );
18. }
19. }
20. export **default** App;

**Output**



## **2. CSS Stylesheet**

You can write styling in a separate file for your React application, and save the file with a .css extension. Now, you can **import** this file in your application.

### **Example**

**App.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
3. **import** './App.css';
5. **class** App **extends** React.Component {
6. render() {
7. **return** (
8. <div>
9. <h1>Hello JavaTpoint</h1>
10. <p>Here, you can find all CS tutorials.</p>
11. </div>
12. );
13. }
14. }
15. export **default** App;

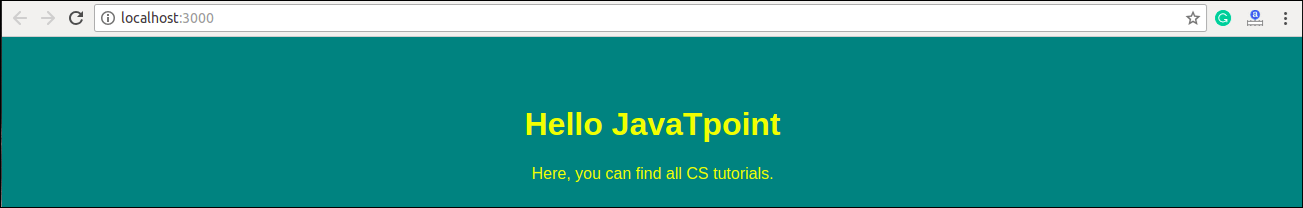
**App.css**

1. body {
2. background-color: #008080;
3. color: yellow;
4. padding: 40px;
5. font-family: Arial;
6. text-align: center;
7. }

**Index.html**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8" />
5. <meta name="viewport"
6. content="width=device-width, initial-scale=1" />
7. <title>React App</title>
8. </head>
9. <body>
10. <div id="app"></div>
11. </body>
12. </html>

**Output**



## **3. CSS Module**

CSS Module is another way of adding styles to your application. It is a **CSS file** where all class names and **animation** names are scoped locally by default. It is available only for the component which imports it, means any styling you add can never be applied to other components without your permission, and you never need to worry about name conflicts. You can create CSS Module with the **.module.css** extension like a **myStyles.module.css** name.

### **Example**

**App.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
3. **import** styles from './myStyles.module.css';
5. **class** App **extends** React.Component {
6. render() {
7. **return** (
8. <div>
9. <h1 className={styles.mystyle}>Hello JavaTpoint</h1>
10. <p className={styles.parastyle}>It provides great CS tutorials.</p>
11. </div>
12. );
13. }
14. }
15. export **default** App;

**myStyles.module.css**

1. .mystyle {
2. background-color: #cdc0b0;
3. color: Red;
4. padding: 10px;
5. font-family: Arial;
6. text-align: center;
7. }
9. .parastyle{
10. color: Green;
11. font-family: Arial;
12. font-size: 35px;
13. text-align: center;
14. }

**Output**



## **4. Styled Components**

Styled-components is a **library** for React. It uses enhance CSS for styling React component systems in your application, which is written with a mixture of JavaScript and CSS.

**The styled-components provides:**

* Automatic critical CSS
* No class name bugs
* Easier deletion of CSS
* Simple dynamic styling
* Painless maintenance

### **Installation**

The styled-components library takes a single command to install in your React application. which is:

1. $ npm install styled-components --save

**Example**

Here, we create a variable by selecting a particular HTML element such as **<div>**, **<Title>**, and **<paragraph>** where we store our style attributes. Now we can use the name of our variable as a wrapper **<Div></Div>** kind of React component.

**App.js**

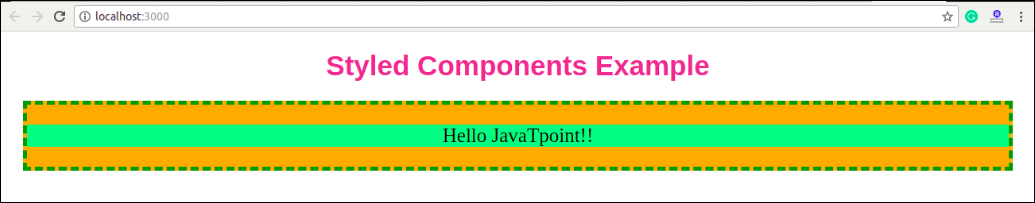
1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
3. **import** styled from 'styled-components';
5. **class** App **extends** React.Component {
6. render() {
7. **const** Div:any = styled.div`
8. margin: 20px;
9. border: 5px dashed green;
10. &:hover {
11. background-color: ${(props:any) => props.hoverColor};
12. }
13. `;
14. **const** Title = styled.h1`
15. font-family: Arial;
16. font-size: 35px;
17. text-align: center;
18. color: palevioletred;
19. `;
20. **const** Paragraph = styled.p`
21. font-size: 25px;
22. text-align: center;
23. background-Color: lightgreen;
24. `;
25. **return** (
26. <div>
27. <Title>Styled Components Example</Title>
28. <p></p>
29. <Div hoverColor="Orange">
30. <Paragraph>Hello JavaTpoint!!</Paragraph>
31. </Div>
32. </div>
33. );
34. }
35. }
36. export **default** App;

**Output**

Now, execute the App.js file, we will get the output as shown below.



When we move the mouse pointer over the image, its color will be changed, as shown in the below image.



Next Topic[React Animation](https://www.javatpoint.com/react-animation)