|  |  |
| --- | --- |
| **ch15.ReactJS Component API** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Component API

ReactJS component is a top-level API. It makes the code completely individual and reusable in the application. It includes various methods for:

* Creating elements
* Transforming elements
* Fragments

Here, we are going to explain the three most important methods available in the React component API.

1. setState()
2. forceUpdate()
3. findDOMNode()

## **setState()**

This method is used to update the state of the component. This method does not always replace the state immediately. Instead, it only adds changes to the original state. It is a primary method that is used to update the user interface(UI) in response to event handlers and server responses.

#### **Note: In the ES6 classes, this.method.bind(this) is used to manually bind the setState() method.**

### **Syntax**

1. **this**.stateState(object newState[, function callback]);

In the above syntax, there is an optional **callback** function which is executed once setState() is completed and the component is re-rendered.

Competitive questions on Structures in Hindi

Keep Watching

Skip Ad

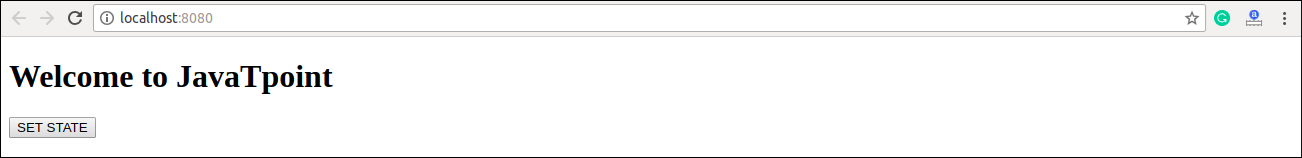
### **Example**

1. **import** React, { Component } from 'react';
2. **import** PropTypes from 'prop-types';
3. **class** App **extends** React.Component {
4. constructor() {
5. **super**();
6. **this**.state = {
7. msg: "Welcome to JavaTpoint"
8. };
9. **this**.updateSetState = **this**.updateSetState.bind(**this**);
10. }
11. updateSetState() {
12. **this**.setState({
13. msg:"Its a best ReactJS tutorial"
14. });
15. }
16. render() {
17. **return** (
18. <div>
19. <h1>{**this**.state.msg}</h1>
20. <button onClick = {**this**.updateSetState}>SET STATE</button>
21. </div>
22. );
23. }
24. }
25. export **default** App;

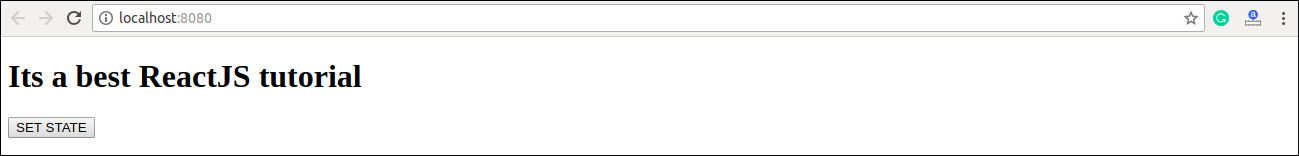
**Main.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
3. **import** App from './App.js';
5. ReactDOM.render(<App/>, document.getElementById('app'));

**Output:**



When you click on the **SET STATE** button, you will see the following screen with the updated message.



## **forceUpdate()**

This method allows us to update the component manually.

### **Syntax**

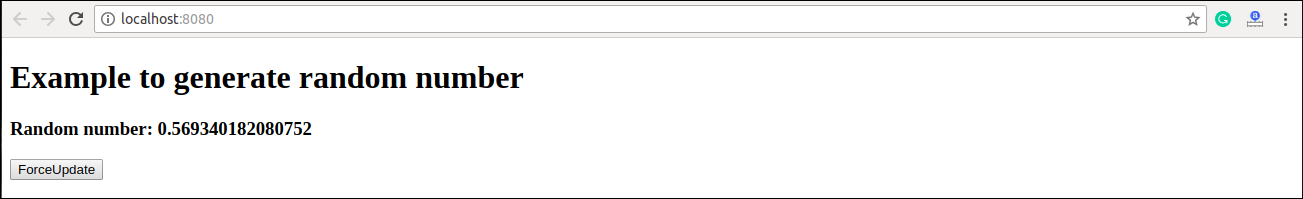
1. Component.forceUpdate(callback);

### **Example**

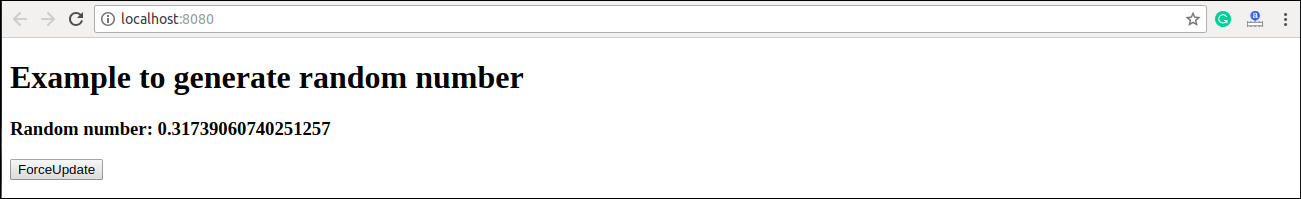
**App.js**

1. **import** React, { Component } from 'react';
2. **class** App **extends** React.Component {
3. constructor() {
4. **super**();
5. **this**.forceUpdateState = **this**.forceUpdateState.bind(**this**);
6. }
7. forceUpdateState() {
8. **this**.forceUpdate();
9. };
10. render() {
11. **return** (
12. <div>
13. <h1>Example to generate random number</h1>
14. <h3>Random number: {Math.random()}</h3>
15. <button onClick = {**this**.forceUpdateState}>ForceUpdate</button>
16. </div>
17. );
18. }
19. }
20. export **default** App;

**Output:**



Each time when you click on **ForceUpdate** button, it will generate the **random** number. It can be shown in the below image.



## **findDOMNode()**

For DOM manipulation, you need to use **ReactDOM.findDOMNode()** method. This method allows us to find or access the underlying DOM node.

### **Syntax**

1. ReactDOM.findDOMNode(component);

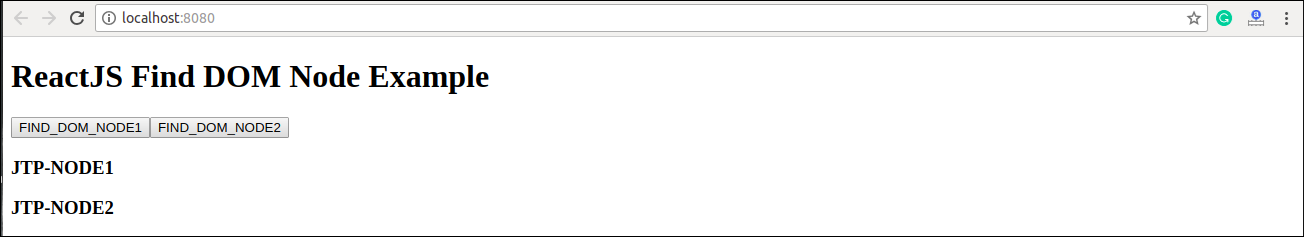
### **Example**

For DOM manipulation, first, you need to import this line: **import ReactDOM** from '**react-dom**' in your **App.js** file.

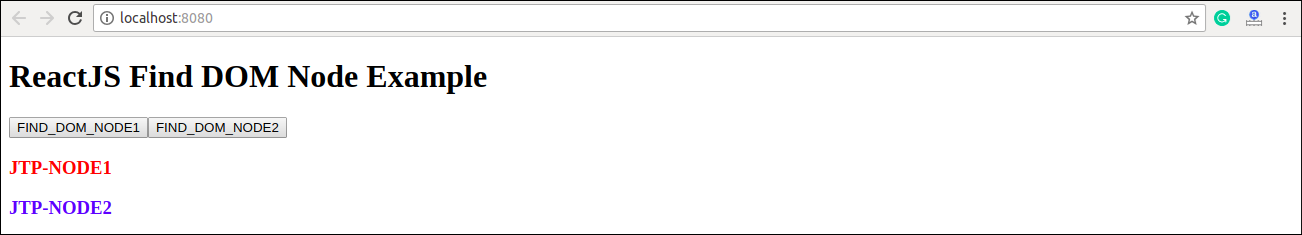
**App.js**

1. **import** React, { Component } from 'react';
2. **import** ReactDOM from 'react-dom';
3. **class** App **extends** React.Component {
4. constructor() {
5. **super**();
6. **this**.findDomNodeHandler1 = **this**.findDomNodeHandler1.bind(**this**);
7. **this**.findDomNodeHandler2 = **this**.findDomNodeHandler2.bind(**this**);
8. };
9. findDomNodeHandler1() {
10. var myDiv = document.getElementById('myDivOne');
11. ReactDOM.findDOMNode(myDivOne).style.color = 'red';
12. }
13. findDomNodeHandler2() {
14. var myDiv = document.getElementById('myDivTwo');
15. ReactDOM.findDOMNode(myDivTwo).style.color = 'blue';
16. }
17. render() {
18. **return** (
19. <div>
20. <h1>ReactJS Find DOM Node Example</h1>
21. <button onClick = {**this**.findDomNodeHandler1}>FIND\_DOM\_NODE1</button>
22. <button onClick = {**this**.findDomNodeHandler2}>FIND\_DOM\_NODE2</button>
23. <h3 id = "myDivOne">JTP-NODE1</h3>
24. <h3 id = "myDivTwo">JTP-NODE2</h3>
25. </div>
26. );
27. }
28. }
29. export **default** App;

**Output:**



Once you click on the **button**, the color of the node gets changed. It can be shown in the below screen.



Next Topic[React Component Life-Cycle](https://www.javatpoint.com/react-component-life-cycle)