|  |  |
| --- | --- |
| **ch10.ReactJS State** | **Date: 22-02-2022** |

**Topics**

React State, Defining State, Update/Change the State

# React State

* The state is an updatable structure that is used to contain data or information about the component.
* The state in a component can change over time. The change in state over time can happen as a response to user action or system event.
* A component with the state is known as stateful components.
* It is the heart of the react component which determines the behavior of the component and how it will render.
* They are also responsible for making a component dynamic and interactive.
* A state must be kept as simple as possible.
* It can be set by using the **setState()** method and calling setState() method triggers UI updates.
* A state represents the component's local state or information. It can only be accessed or modified inside the component or by the component directly.
* To set an initial state before any interaction occurs, we need to use the **getInitialState()** method.
* **For example**, if we have five components that need data or information from the state, then we need to create one container component that will keep the state for all of them.

# Defining State

To define a state, you have to first declare a default set of values for defining the component's initial state. To do this, add a class constructor which assigns an initial state using this.state. The '**this.state**' property can be rendered inside **render()** method.

# Example

The below sample code shows how we can create a stateful component using ES6 syntax.

import React, { Component } from 'react';

class App extends *React*.Component {

 constructor() {

      super();

*this*.state = { displayBio: true };

      }

      render() {

          const bio = *this*.state.displayBio ? (

              <div>

                  <p><h3>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad. We have a team of experienced Java developers and trainers from multinational companies to teach our campus students.</h3></p>

            </div>

              ) : null;

              return (

                  <div>

                      <h1> Welcome to JavaTpoint!! </h1>

                      { bio }

                  </div>

              );

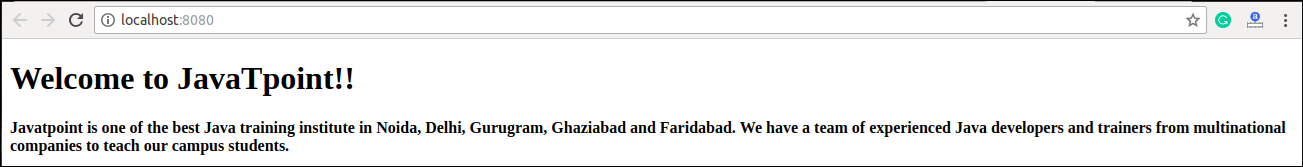
     }

}

export default App;

To set the state, it is required to call the super() method in the constructor. It is because this.state is uninitialized before the super() method has been called.

**Output**



# Changing the State

We can change the component state by using the setState() method and passing a new state object as the argument. Now, create a new method toggleDisplayBio() in the above example and bind this keyword to the toggleDisplayBio() method otherwise we can't access this inside toggleDisplayBio() method.

**this**.toggleDisplayBio = **this**.toggleDisplayBio.bind(**this**);

# Example

In this example, we are going to add a **button** to the **render**() method. Clicking on this button triggers the toggleDisplayBio() method which displays the desired output.

import React, { Component } from 'react';

class App extends *React*.Component {

 constructor() {

      super();

*this*.state = { displayBio: false };

      console.log('Component this', *this*);

*this*.toggleDisplayBio = *this*.toggleDisplayBio.bind(*this*);

      }

      toggleDisplayBio(){

*this*.setState({displayBio: !*this*.state.displayBio});

          }

      render() {

          return (

              <div>

                  <h1>Welcome to JavaTpoint!!</h1>

                  {

*this*.state.displayBio ? (

                          <div>

                              <p><h4>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad. We have a team of experienced Java developers and trainers from multinational companies to teach our campus students.</h4></p>

                              <button *onClick*={*this*.toggleDisplayBio}> Show Less </button>

                        </div>

                          ) : (

                              <div>

                                  <button *onClick*={*this*.toggleDisplayBio}> Read More </button>

                              </div>

                          )

                  }

             </div>

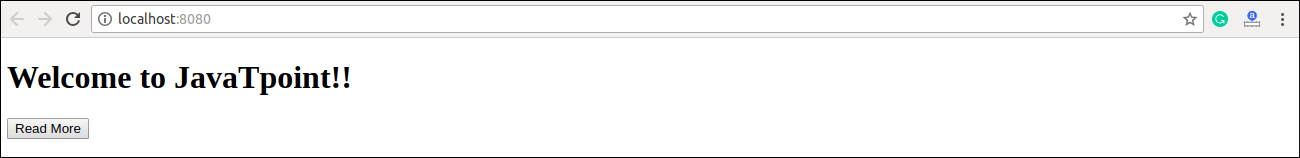
        )

    }

}

export default App;

**Output:**



When you click the **Read More** button, you will get the below output, and when you click the **Show Less** button, you will get the output as shown in the above image.

