**Source -geeksforgeeks**

**ReactJS Forwarding Refs**

The forwardRef method in React allows parent components to move down (or “forward”) refs to their children. ForwardRef gives a child component a reference to a DOM entity created by its parent component in React. This helps the child to read and modify the element from any location where it is used.

**How does forwardRef work in React?**

In React, parent components typically use props to transfer data down to their children. Consider you make a child component with a new set of props to change its behavior. We need a way to change the behavior of a child component without having to look for the state or re-rendering the component. We can do this by using refs. We can access a DOM node that is represented by an element using refs. As a result, we will make changes to it without affecting its state or having to re-render it.

When a child component needs to refer to its parent’s current node, the parent component must have a way for the child to receive its ref. The technique is known as ref forwarding.

**Syntax:**

React.forwardRef((props, ref) => {})

**Paramters:** It takes a function with props and ref arguments.

**Return Value:** This function returns a JSX Element.

**Example:**Now write down the following code in the **App.js** file. Here, App is our default component where we have written our code.

* App.js

|  |
| --- |
| import React from 'react'    class App extends React.Component {    constructor(props) {      super(props)      this.aRef = React.createRef()    }    render() {      return (        <>          <Counter ref={this.aRef} />          <button onClick={() => { console.log(this.aRef) }}>           Ref          </button>        </>      )    }  }    const Counter = React.forwardRef((props, ref) => {    class Counter extends React.Component {      constructor(props) {        super(props)        this.state = {          count: 0        }      }      render() {        return (          <div>            Count: {this.state.count}            <button ref={ref} onClick={() => this.setState(              { count: this.state.count + 1 })}>                    Increment            </button>          </div>        )      }    }    return <Counter />  })    export default App |

**Explanation:**Since the counter is within the function, it can access the props and ref parameters using a closure. The counter is rendered and returned. The ref passed to the Counter component is set to the value of the button element’s ref attribute. The counter element’s ref attribute will now be set to refer to the button element.

The Counter component is rendered by the App component. It starts by creating a ref *this.aRef*is assigned to the Counter component’s ref attribute as a value. We’ve included a button that logs the value of *this.aRef*. The *this.aRef* will hold the HTMLButtonElement of the Incr button in the Counter component. Clicking on the Ref button will confirm that It will log *this.aRef* which will log the object of the Incr button HTMLButtonElement. It didn’t log the instance of the Counter because the Counter component forwarded it to its child component, the Increment button.

