|  |  |
| --- | --- |
| **ch39.ReactJS Portals** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# React Portals

The **React 16.0** version introduced React portals in **September 2017**. A React portal provides a way to render an element outside of its component hierarchy, i.e., in a separate component.

Before React 16.0 version, it is very tricky to render the child component outside of its parent component hierarchy. If we do this, it breaks the convention where a component needs to render as a new element and follow a **parent-child** hierarchy. In React, the parent component always wants to go where its child component goes. That's why React portal concept comes in.

### **Syntax**

1. ReactDOM.createPortal(child, container)

Here, the first argument (child) is the component, which can be an element, string, or fragment, and the second argument (container) is a DOM element.

### **Example before React v16**

Generally, when you want to return an element from a component's render method, it is mounted as a new div into the DOM and render the children of the closest parent component.

1

1. render() {
2. // React mounts a new div into the DOM and renders the children into it
3. **return** (
4. <div>
5. {**this**.props.children}
6. </div>
7. );
8. }

### **Example using portal**

But, sometimes we want to insert a child component into a different location in the DOM. It means React does not want to create a new div. We can do this by creating React portal.

1. render() {
2. **return** ReactDOM.createPortal(
3. **this**.props.children,
4. myNode,
5. );
6. }

## **Features**

* It uses React version 16 and its official API for creating portals.
* It has a fallback for React version 15.
* It transports its children component into a new React portal which is appended by default to document.body.
* It can also target user specified DOM element.
* It supports server-side rendering
* It supports returning arrays (no wrapper div's needed)
* It uses <Portal /> and <PortalWithState /> so there is no compromise between flexibility and convenience.
* It doesn't produce any DOM mess.
* It has no dependencies, minimalistic.

## **When to use?**

The common use-cases of React portal include:

* Modals
* Tooltips
* Floating menus
* Widgets

## **Installation**

We can install React portal using the following command.

1. $ npm install react-portal --save

## **Explanation of React Portal**

Create a new React project using the following command.

1. $ npx create-react-app reactapp

Open the App.js file and insert the following code snippet.

**App.js**

1. **import** React, {Component} from 'react';
2. **import** './App.css'
3. **import** PortalDemo from './PortalDemo.js';
5. **class** App **extends** Component {
6. render () {
7. **return** (
8. <div className='App'>
9. <PortalDemo />
10. </div>
11. );
12. }
13. }
14. export **default** App;

The next step is to create a **portal** component and import it in the App.js file.

**PortalDemo.js**

1. **import** React from 'react'
2. **import** ReactDOM from 'react-dom'
4. function PortalDemo(){
5. **return** ReactDOM.createPortal(
6. <h1>Portals Demo</h1>,
7. document.getElementById('portal-root')
8. )
9. }
10. export **default** PortalDemo

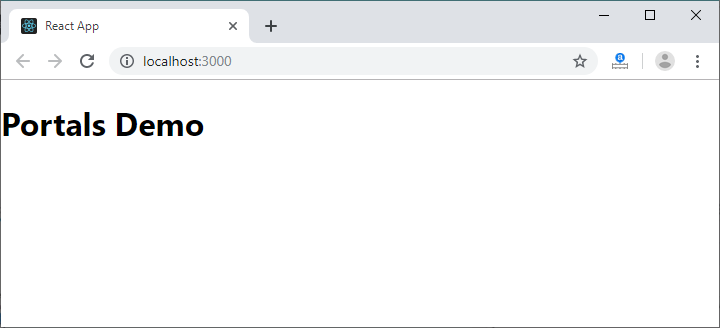
Now, open the Index.html file and add a <div id="portal-root"></div> element to access the child component outside the root node.

**Index.html**

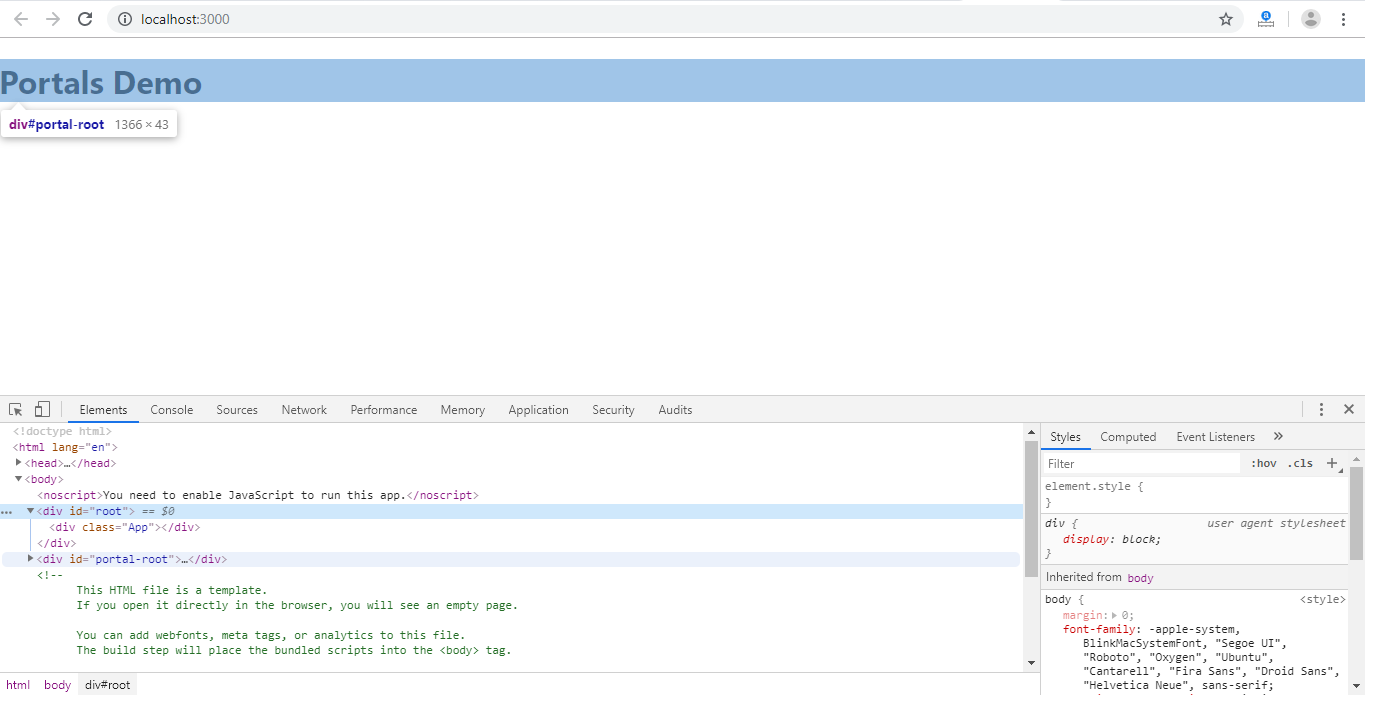
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8" />
5. <link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico" />
6. <meta name="viewport" content="width=device-width, initial-scale=1" />
7. <meta name="theme-color" content="#000000" />
8. <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />
9. <title>React App</title>
10. </head>
11. <body>
12. <noscript>It is required to enable JavaScript to run **this** app.</noscript>
13. <div id="root"></div>
14. <div id="portal-root"></div>
15. </body>
16. </html>

**Output:**

When we execute the React app, we will get the following screen.



Now, open the **Inspect** (ctrl + shift + I). In this window, select the **Elements** section and then click on the <div id="portal-root"></div> component. Here, we can see that each tag is under the "portal-root" DOM node, not the "root" DOM node. Hence, we can see that how React Portal provides the ability to break out of root DOM tree.



Next Topic[React Error Boundaries](https://www.javatpoint.com/react-error-boundaries)