Портали

<https://reactjs.org/docs/portals.html>

Портали дозволяють монтувати компоненти у вузли DOM, що не є дочірніми стосовно батьківського компонента. Не дивлячись на те, що дочірні компоненти будуть знаходитись не у дочірніх вузлах DOM дерева, вони поводяться так, ніби знаходяться у дочірньому вузлі (аналіз подій, підняття подій і т.д.).

|  |  |
| --- | --- |
|  |  |
| render() {  ReactDOM.createPortal(  дочірні компоненти,  контейнер)  } | render() {  // React does \*not\* create a new div. It renders the children into `domNode`.  // `domNode` is any valid DOM node, regardless of its location in the DOM.  return **ReactDOM.createPortal**(  this.props.children,  domNode  );  } |

Створити модальне вікно

<https://codepen.io/gaearon/pen/yzMaBd>

|  |  |
| --- | --- |
|  | #modal-root {  position: relative;  z-index: 999;  }  .app {  height: 10em;  width: 10em;  background: lightblue;  overflow: hidden;  }  .modal {  background-color: rgba(0,0,0,0.5);  position: fixed;  height: 100%;  width: 100%;  top: 0;  left: 0;  display: flex;  align-items: center;  justify-content: center;  } |
|  | //Елементи, у які буде робитися вставка елементів  <div id="app-root"></div>  **<div id="modal-root"></div>** |
|  | // These two containers are siblings in the DOM  const appRoot = document.getElementById('app-root');  const modalRoot = document.getElementById('modal-root');  // Let's create a Modal component that is an abstraction around  // the portal API.  class Modal extends React.Component {  constructor(props) {  super(props);  // Create a div that we'll render the modal into. Because each  // Modal component has its own element, we can render multiple  // modal components into the modal container.  this.el = document.createElement('div');  }  componentDidMount() {  // Append the element into the DOM on mount. We'll render  // into the modal container element (see the HTML tab).  modalRoot.appendChild(this.el);  }  componentWillUnmount() {  // Remove the element from the DOM when we unmount  modalRoot.removeChild(this.el);  }    render() {  // Use a portal to render the children into the element  **return ReactDOM.createPortal(**  // Any valid React child: JSX, strings, arrays, etc.  this.props.children,  // A DOM element  this.el,  );  }  }  // The Modal component is a normal React component, so we can  // render it wherever we like without needing to know that it's  // implemented with portals.  class App extends React.Component {  constructor(props) {  super(props);  this.state = {showModal: false};    this.handleShow = this.handleShow.bind(this);  this.handleHide = this.handleHide.bind(this);  }  handleShow() {  this.setState({showModal: true});  }    handleHide() {  this.setState({showModal: false});  }  render() {  // Show a Modal on click.  // (In a real app, don't forget to use ARIA attributes  // for accessibility!)  const modal = this.state.showModal ? (  <Modal>  <div className="modal">  <div>  With a portal, we can render content into a different  part of the DOM, as if it were any other React child.  </div>  This is being rendered inside the #modal-container div.  <button onClick={this.handleHide}>Hide modal</button>  </div>  </Modal>  ) : null;  return (  <div className="app">  This div has overflow: hidden.  <button onClick={this.handleShow}>Show modal</button>  {modal}  </div>  );  }  }  ReactDOM.render(<App />, appRoot); |
|  |  |

Приклад 2.

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | **import** React, {PureComponent,Fragment} **from 'react'**; **import** ReactDOM **from 'react-dom'**;  **class** TestComp **extends** PureComponent {  constructor(props)  {  **super**(props);   **this**.**div**=**document**.createElement(**"div"**);  **this**.**div**.setAttribute(**"id"**,**"mod"**);   **let** inner=**document**.createElement(**"div"**);  inner.setAttribute(**"id"**,**"inner"**);  **this**.**div**.appendChild(inner);  **document**.**body**.appendChild(**this**.**div**);   }  componentWillUnmount(){  **document**.**body**.removeChild(**this**.**div**);  }   render() {  **const** {onclick} = **this**.**props**;  **return** ReactDOM.createPortal(  <**Fragment**>  {**this**.**props**.**children**}  <**button onClick=**{**this**.**props**.*hide*}> Close</**button**>,   </**Fragment**>,  **document**.getElementById(**"inner"**)  )  }; }  **export default** TestComp; | **#mod**{  **position**: **fixed**;  **left**: 0%;  **top**:0%;  **width**:100%;  **height**: 100%;  **background**: **gray**;  **opacity**: 0.5;  z-index:999 } **#inner**{  **position**: **fixed**;  **left**:20%;  **top**:20%;  **border**:2**px solid black**;  **background**: **white**;  **opacity**: 1; }  -----------------------  <body>  . . . . . .  <div id=”mod”>  <div id=”inner”>    </div>  </div>  </body> |
|  | **import** React, {Component} **from 'react'**; **import** PropTypes **from 'prop-types'**; **import** TestComponent **from "./TestComp" class** MyComponent **extends** Component {  constructor(props){  **super**(props);  **this**.**state**={  **showModal**:**true** }  }  *hide*=()=>{  **this**.setState({  **showModal**:**false** })  }  *show*=()=>{  **this**.setState({  **showModal**:**true** })  }  render(){  **return**(  <**div**>  {**this**.**state**.**showModal**&& <**TestComponent hide=**{**this**.*hide*}> <**p**>Hello</**p**></**TestComponent**>}  <**button onClick=**{**this**.*show*}>Show</**button**>  </**div**>  )  } }    **export default** MyComponent; | |
|  | **import** React, { Component } **from 'react'**; **import** logo **from './logo.svg'**; **import './App.css'**; **import** TestContainer **from "./TestContainer" class** App **extends** Component {  render() {  **return** (  <**div className="App"**>  <**TestContainer**/>  </**div**>  );  } }  **export default** App | |