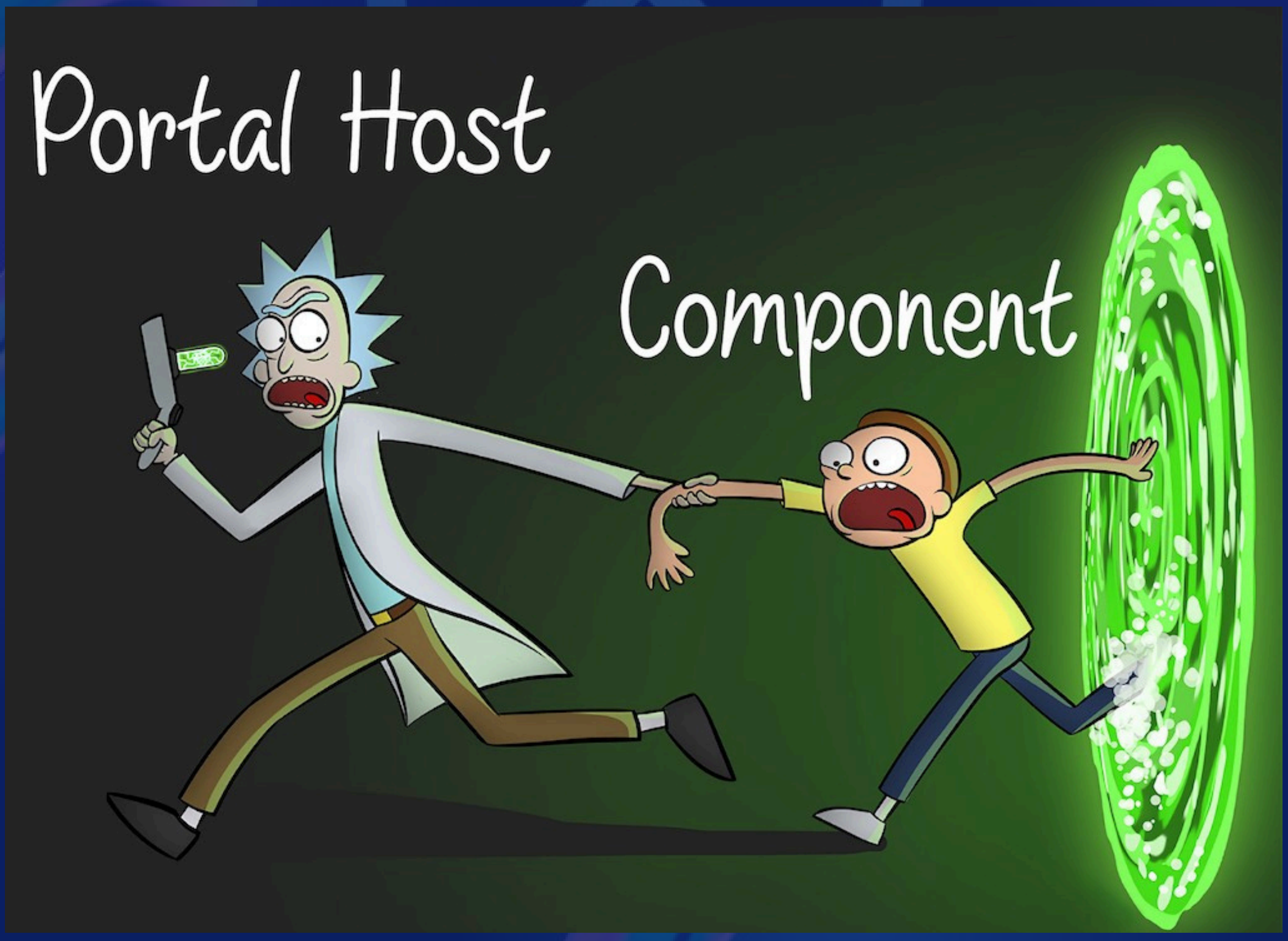
Day 29

Portals In React



@oluwakemi Oluwadahunsi

Introduction

React Portals are a powerful feature that allows you to render components **outside** the normal **DOM** hierarchy.

This can be extremely useful in certain scenarios, such as when you need to **render modals**, **tooltips**, **popups**, or other elements that must visually and structurally break out of their **parent component's** confines.

What is a Portal in React?

By default, a **React component** is rendered as a child of its parent component in the DOM.

However, there are times when you need a component to be rendered somewhere else in the DOM tree, **outside of its parent**.

This is where React Portals come in.

@oluwakemi Oluwadahunsi

They allow you to render a **child component** into a **DOM node** that exists **outside** the **DOM hierarchy** of the **parent component**.

To better understand this, consider the structure of a typical React application where components are nested within other components. Usually, the DOM structure looks something like this:

```
<div id="root">
  <div>
    <!-- Parent component -->
    <div>
      <!-- Child component -->
    </div>
  </div>
</div>
```

With React Portals, we can render a child component directly into another part of the DOM (for example, a modal or popup that exists outside of the parent container).

@oluwakemi Oluwadahunsi

When Should You Use React Portals?

React Portals are most commonly used in the following situations:

- **Modals and Dialogs:** When you want a modal to appear over the entire page without being constrained by the parent component's styles or z-index issues.
- **Tooltips and Popovers:** For floating elements that need to appear on top of other components without being affected by the parent's layout.
- **Global Notifications:** Alerts, toast notifications, or banners that should appear at the top level of your app.
- **Fixed Position Elements:** Any element that requires fixed or absolute positioning at the document level instead of relative to its parent.

These use cases often require breaking out of the DOM tree, which React Portals elegantly handle.

@oluwakemi Oluwadahunsi

How to Create a Portal in React

Creating a portal in React is straightforward. The process involves two main steps:

Step 1 - Identify a target DOM node outside your component's tree:

This is the node where you want to render the component (e.g., the body tag or a specific div).

Step 2 - Use ReactDOM.createPortal:

This method is used to render the component outside the parent hierarchy.

Example: Creating a Modal with Portals

Here’s an example of creating a simple modal using a React Portal:

First, we need to set up our **HTML file (index.html)** by creating a specific **DOM node** where the modal will be rendered:

```
1 <!— index.html —>
2 <body>
3   <div id="root"></div>
4   <div id="modal-root"></div> <!— This is where the modal will be rendered —>
5 </body>
6
```

another div created in the DOM node

Next, we create the **React component** for the modal and use the **ReactDOM.createPortal** function to render the modal into the **#modal-root** element:

Create a component named “**Modal.jsx**” in your “**src**” folder:

```jsx
import React from 'react';
import ReactDOM from 'react-dom';

const Modal = ({ isOpen, onClose, children }) => {
  if (!isOpen) return null; // If the modal is not open, don't render anything

  return ReactDOM.createPortal(
    <div style={modalStyle}>
      <div style={overlayStyle}>
        <div style={contentStyle}>
          <button onClick={onClose}>Close Modal</button>
          {children}
        </div>
      </div>,
      document.getElementById('modal-root') // Renders inside modal-root
  );
};

// Inline styles for simplicity
const modalStyle = {
  position: 'fixed',
  top: 0,
  left: 0,
  width: '100%',
  height: '100%',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',
  backgroundColor: 'rgba(0, 0, 0, 0.5)'
};

const overlayStyle = {
  backgroundColor: 'white',
  padding: '20px',
  borderRadius: '8px',
  textAlign: 'center'
};

const contentStyle = {
  marginTop: '10px'
};

export default Modal;
```

using ReactDom.createPortal

the root HTML element id from index.html

stylings to make modal visible

- Finally, you can use the Modal component in your **App.jsx** like so:

```jsx
import React, { useState } from 'react';
import Modal from './Modal';

const App = () => {
  const [isModalOpen, setModalOpen] = useState(false);

  const openModal = () => setModalOpen(true);
  const closeModal = () => setModalOpen(false);

  return (
    <div>
      <h1>React Portals Example</h1>
      <button onClick={openModal}>Open Modal</button>
      <Modal isOpen={isModalOpen} onClose={closeModal}>
        <h2>This is a modal!</h2>
      </Modal>
    </div>
  );
};

export default App;
```

In this example:

- The modal is rendered using **ReactDOM.createPortal** inside the **#modal-root div** in the DOM.
- The Modal component is only rendered when **isModalOpen** is true.
- We close the modal by passing an **onClose** function to the Modal component.

How ReactDOM.createPortal Works

The ReactDOM.createPortal method takes two arguments:

1. **The JSX to render** (in this case, the modal's JSX).

2. **The DOM node where you want to render it** (in our example, the **#modal-root** div).

The returned JSX is then rendered in the specified DOM node, bypassing the normal DOM hierarchy.

@oluwakemi Oluwadahunsi

Important Notes About React Portals

1. Event Bubbling:

Even though the JSX is rendered outside of its parent in the DOM tree, event bubbling in React works as if the component was still in the same place.

his means that events inside a portal component will propagate up to its parent component (inside the #root element) as if the portal had been rendered there.

2. CSS Positioning Issues:

Portals are a great solution for dealing with CSS issues like z-index conflicts or overflow issues (e.g., a modal inside a container with overflow: hidden would be clipped, but a portal would avoid this problem).

@oluwakemi Oluwadahunsi

Benefits of Using React Portals

1. Solves Overflow and z-index Issues: By rendering outside the parent container, you can avoid common problems like content being cut off or hidden due to parent overflow or **z-index** properties.

2. Improved Flexibility: You can build more flexible components such as modals, tooltips, and popovers without worrying about the component's position in the DOM hierarchy.

3. Better UI Separation: Portals allow you to separate your UI concerns, rendering some UI elements (like modals) at a higher level in the DOM while keeping their logic encapsulated.

@oluwakemi Oluwadahunsi

I hope you found this material useful and helpful.

Remember to:

Like

Save for future reference

&

Share with your network, be helpful to someone 👌

@oluwakemi Oluwadahunsi

Hi There!

Thank you for reading through

Did you enjoy this knowledge?

💼 Follow my LinkedIn page for more work-life balancing and Coding tips.

🌐 LinkedIn: Oluwakemi Oluwadahunsi

kodemaven-portfolio.vercel.app