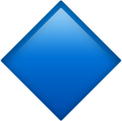
ReactJS Part-2 - Lab Assignment

Shivam Rane – 23BPS1010

# Creating and Using Class Components with Constructors

 **Concepts Covered**: Class Components, Constructors, State Initialization

 **Task**:

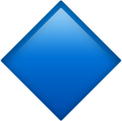
* + Create a **class component** with a constructor that initializes state.
  + Display a **welcome message** with the user's name stored in the state.



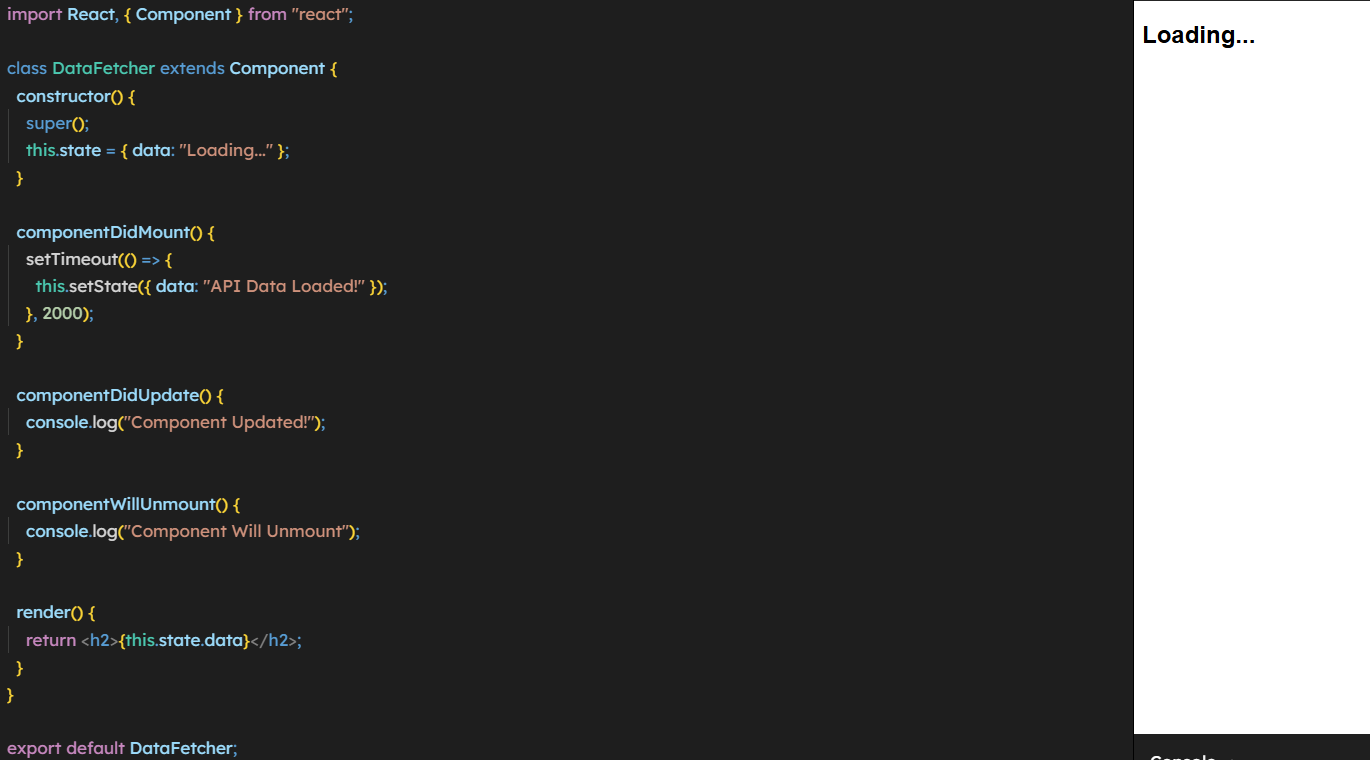
# Implementing Component Life Cycle Methods

 **Concepts**

**Covered**: componentDidMount, componentDidUpdate, componentWillUnmount

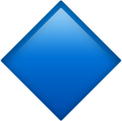
 **Task**:

* + Create a class component that **fetches data from an API** in componentDidMount.
  + Update the state when the user clicks a button (componentDidUpdate).
  + Cleanup when the component is unmounted (componentWillUnmount).

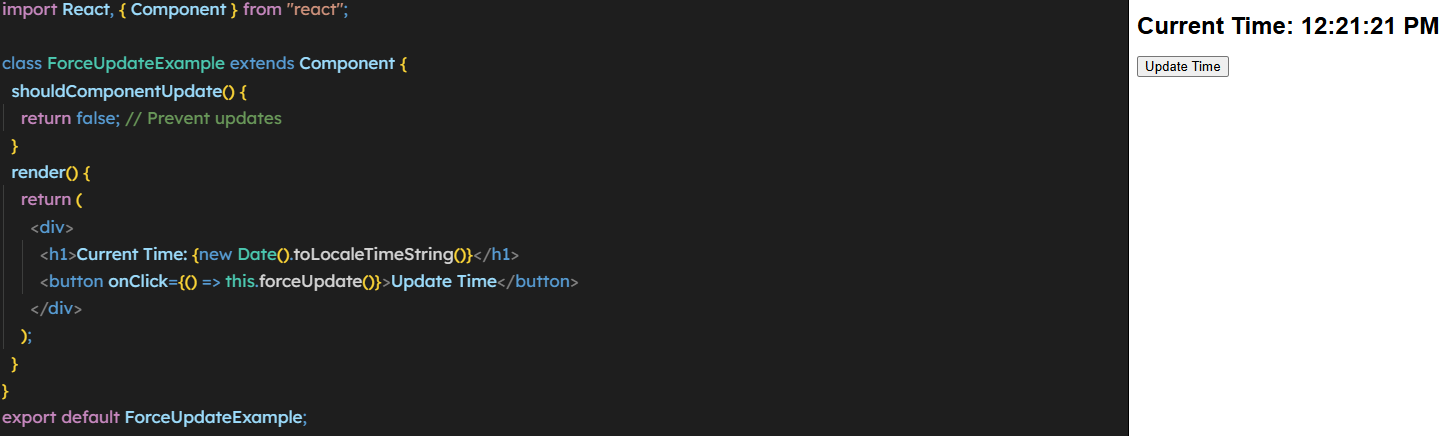


1. **Using React Component API: forceUpdate and shouldComponentUpdate**

 **Concepts Covered**: forceUpdate, shouldComponentUpdate

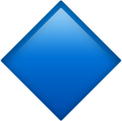
 **Task**:

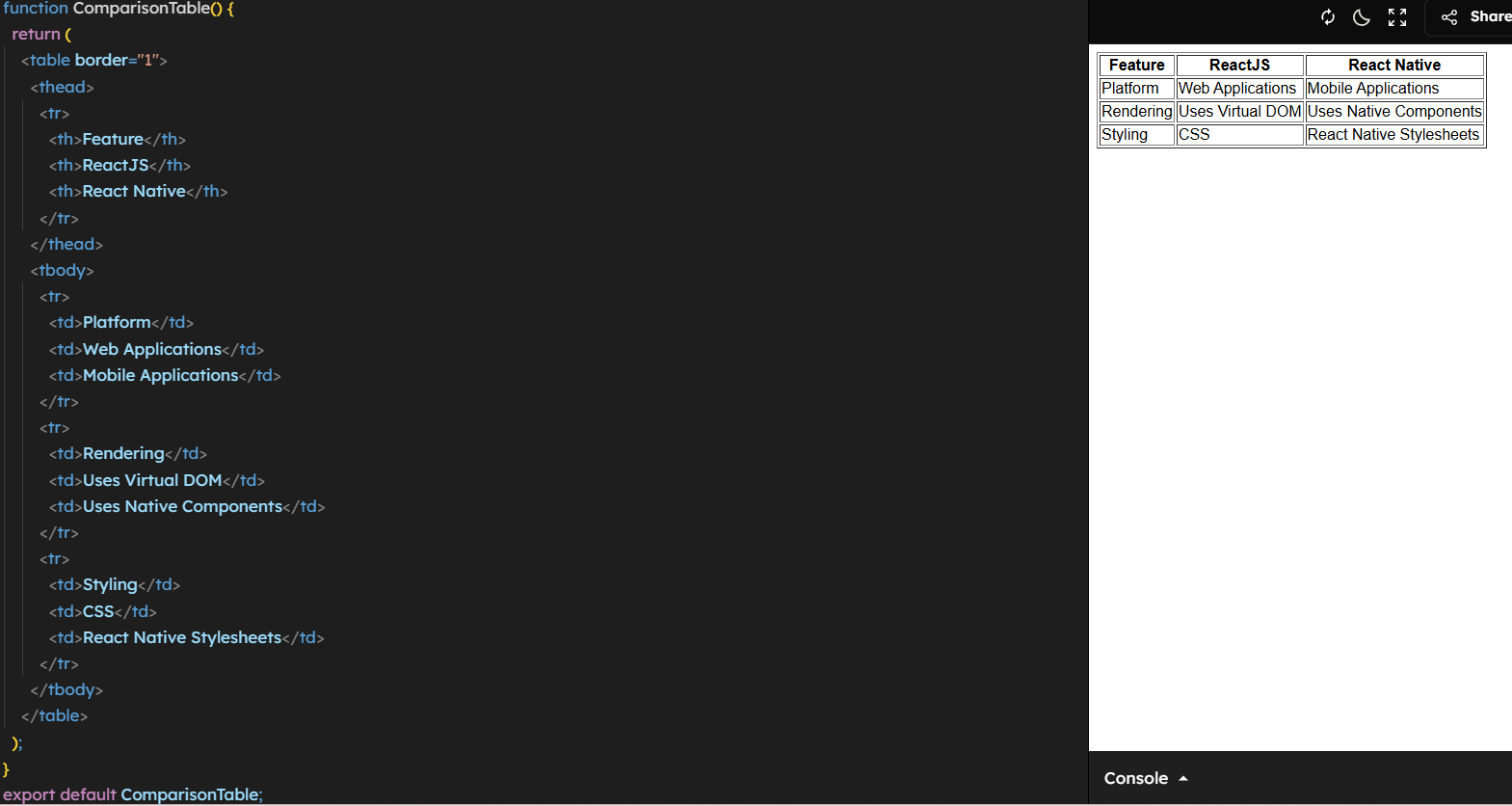
* + Create a class component that **prevents unnecessary updates** using shouldComponentUpdate.
  + Use a button to **force update** the component.



# Comparing React Native and ReactJS

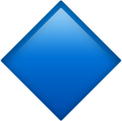
 **Concepts Covered**: Differences Between ReactJS and React Native

 **Task**:

* 1. Write a **table comparison** between ReactJS and React Native.
  2. Create a **React component** that displays this comparison.

# Creating a Parent-Child Component Structure

 **Concepts Covered**: Props, Parent-Child Communication

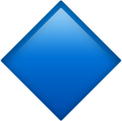
 **Task**:

* 1. Create a **Parent Component** that passes **data** to a **Child Component** via props.

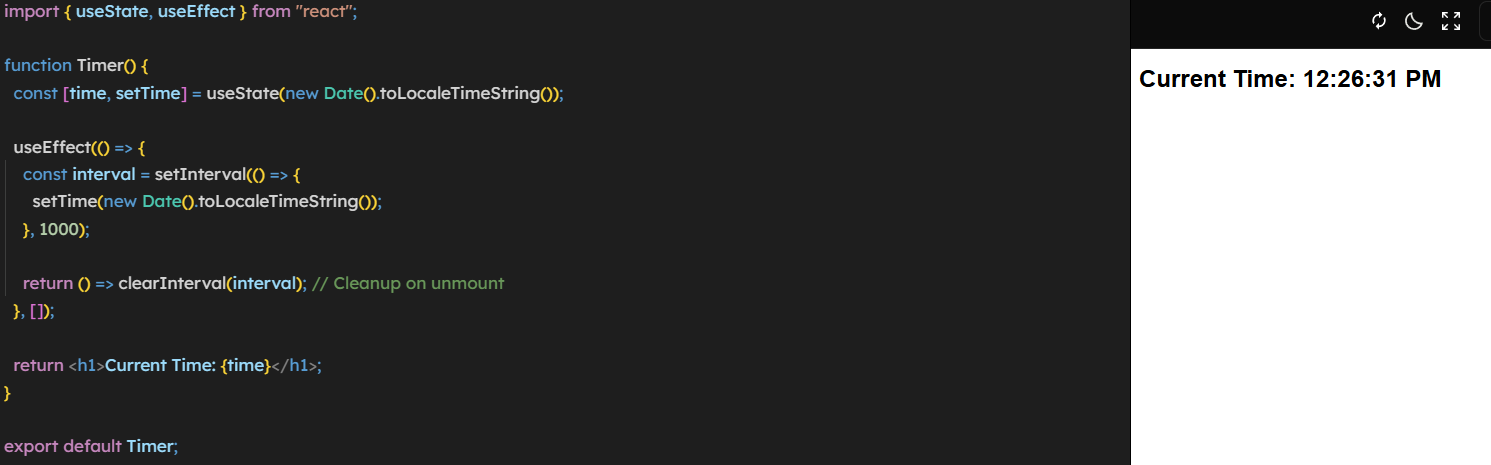


# Managing State and Lifecycle with Hooks (useEffect)

 **Concepts Covered**: React Hooks, useEffect Lifecycle

 **Task**:

* 1. Convert a class component with lifecycle methods into a **functional component using hooks**.



# Implementing Component Composition with Multiple Components

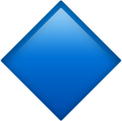
 **Concepts Covered**: Component Reusability, Composition

* **Task**:
  + Create **Header, Content, and Footer** components.
  + Render them inside an **App component**.



# Simulating an API Call and Displaying Data

 **Concepts Covered**: Fetching Data in React

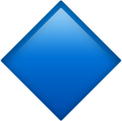
 **Task**:

* 1. Use fetch() or axios to get data from an API.
  2. Display the fetched data in a React component.



# Creating a Component with Controlled Inputs

 **Concepts Covered**: Handling User Input, State Management

 **Task**:

* 1. Create a form component with an **input field** and a **button**.
  2. Update state when the user types.

