Exercise 15: React Hook (useReducer)

Objectives and Outcomes

useReducer is a React hook that allows you to manage state and state transitions within a functional component. It is an alternative to using the useState hook when you have more complex state logic that involves multiple actions. These exercises will help you understand how to use useContext in React

Exercises

1. Use useReducer to manage state in a simple counter component.

Define a simple counterReducer function that takes the current state and an action as parameters. The reducer function evaluates the action.type using a switch statement and performs the corresponding state update.

The Counter component uses the useReducer hook to initialize the count state to 0 and provide the counterReducer as the reducer function. The useReducer hook returns the current state (count) and a dispatch function to dispatch actions to update the state.

Inside the Counter component, we render the current count using JSX. The "+" and "-" buttons dispatch the "INCREMENT" and "DECREMENT" actions, respectively, when clicked. The "Reset" button dispatches the "RESET" action.

Each button's onClick event handler calls the dispatch function with the corresponding action object. The counterReducer function is then invoked and updates the state based on the dispatched action.

1. Create a Question Bank.

Define an initialState object that holds the initial values for the state variables.

const initialState = {

questions: [

{

id: 1,

question: 'What is the capital of Australia?',

options: ['Sydney', 'Canberra', 'Melbourne', 'Perth'],

answer: 'Canberra',

},

{

id: 2,

question: 'Which planet is known as the Red Planet?',

options: ['Venus', 'Mars', 'Jupiter', 'Saturn'],

answer: 'Mars',

},

// Add more questions here

// ...

],

currentQuestion: 0,

selectedOption: '',

score: 0,

showScore: false,

};

We also define a reducer function that takes the current state and an action as parameters. The reducer function evaluates the action.type using a switch statement and performs the corresponding state update.

case 'SELECT\_OPTION': // Add more code here

case 'NEXT\_QUESTION': // Add more code here

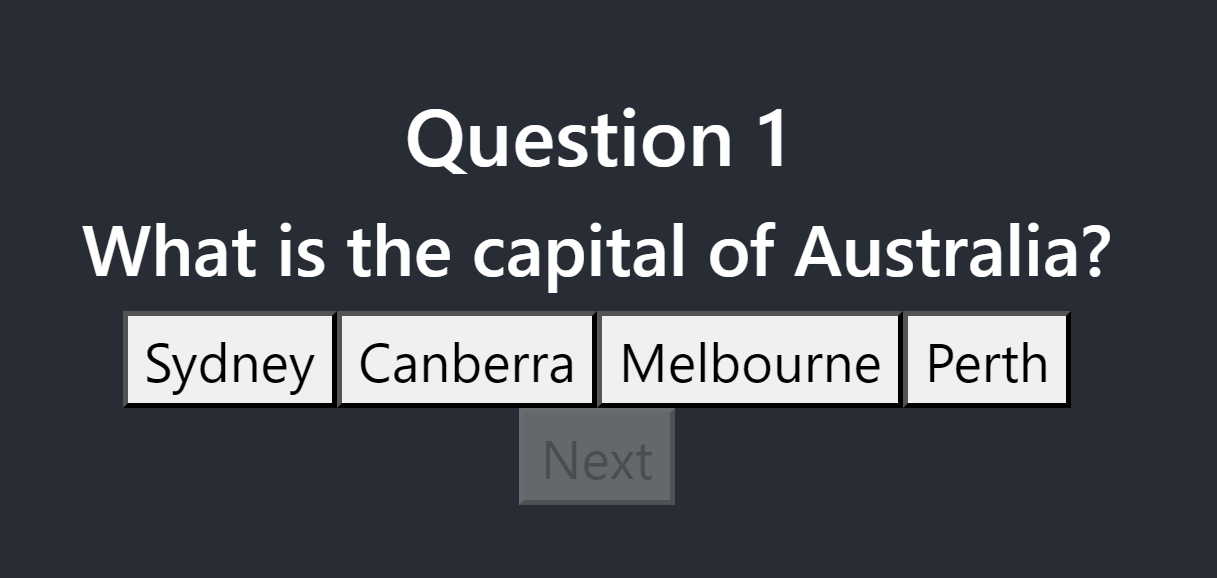
case 'RESTART\_QUIZ': // Add more code here

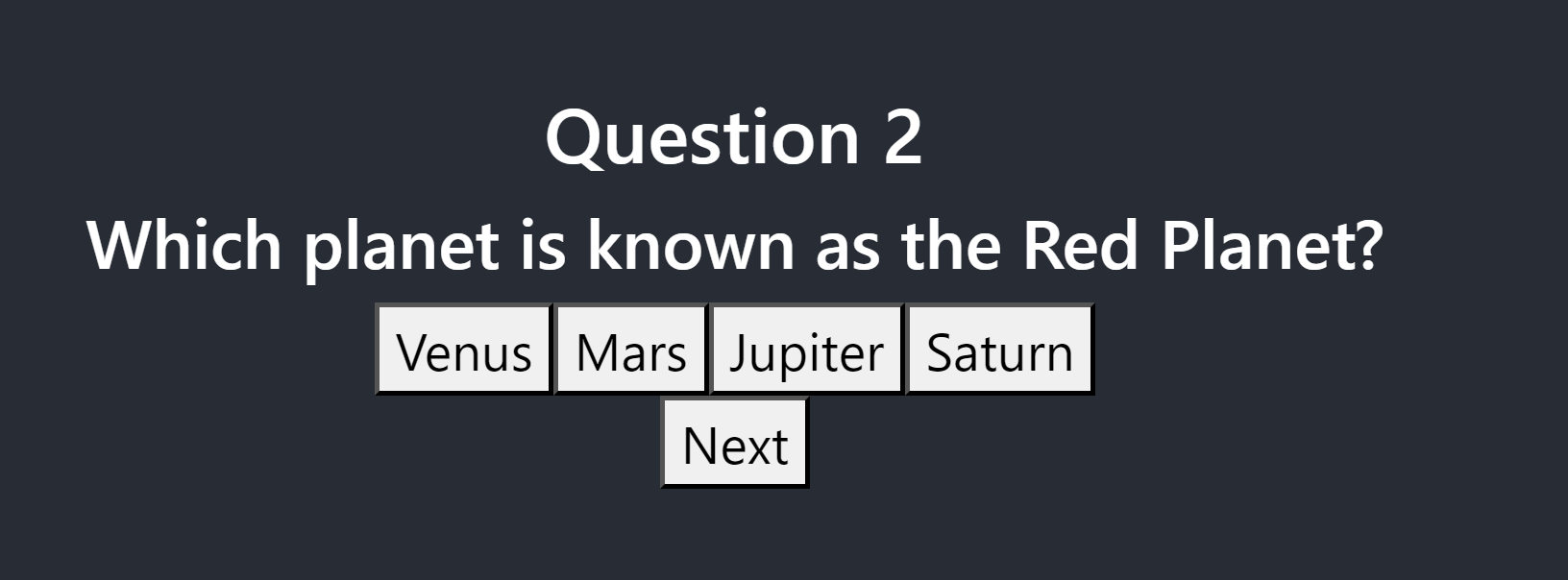
The QuestionBank component now uses the useReducer hook, passing in the reducer function and initialState as arguments. The hook returns the current state and a dispatch function to dispatch actions to update the state.

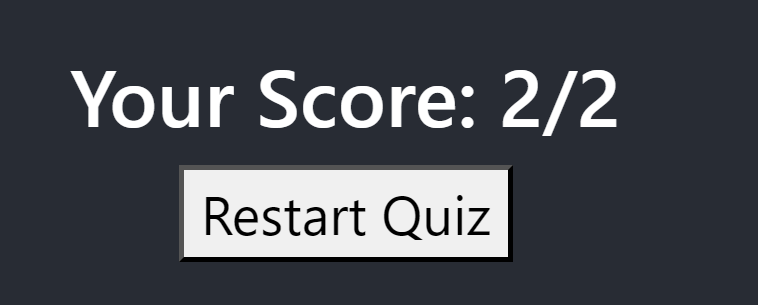
Inside the component, we access the state variables and dispatch actions accordingly. The handleOptionSelect function dispatches the SELECT\_OPTION action with the selected option as the payload. The handleNextQuestion function dispatches the NEXT\_QUESTION action, and the handleRestartQuiz function dispatches the RESTART\_QUIZ action.

The rendering part of the component remains the same, utilizing the state variables from the state object. The actions trigger state updates through the dispatched actions.

Using useReducer provides a more structured approach for managing complex state and state transitions in the question bank component.







Conclusion

In conclusion, the useReducer hook in React provides a way to manage state and state transitions within functional components.