Debugging in Redux

Learning Objectives

By the end of this lesson, you will be able to:

- List the debugging techniques to output information at different points in the code
- Identify the different debugging scenarios to ensure targeted and effective issue resolution
- Compute and debug a React application that consumes a REST API by strategically placing breakpoints using React Developer Tools
- Implement the functionality for storing and retrieving employee details from a JSON file using Axios

Introduction to Debugging in Redux

What Is Debugging?

It is the process of identifying and resolving issues or errors in software code.

It involves tracing and fixing unexpected behaviors, ensuring the smooth operation of the Redux-powered application.

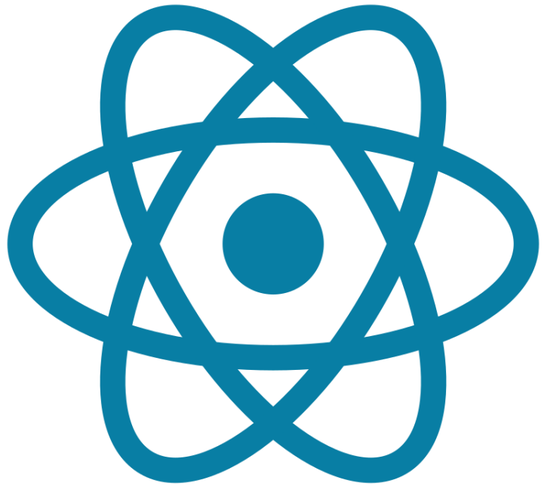
Debugging Techniques

To establish a foundation for debugging in Redux, developers employ various techniques, such as:

- Using console logging to output information at different points in the code
- Setting breakpoints to pause execution for closer inspection
- Inspecting variables to understand their values during runtime

Debugging React Components

React Developer Tools is a browser extension that is primarily focused on assisting developers in understanding and debugging React components.



It provides the following features:

- Inspecting the component hierarchy
- Viewing component state and properties
- Tracking component updates
- Interacting with the Document Object Model (DOM) rendered by React components

It helps with component-level debugging and optimization.

Redux Debugging Tools

They are instrumental in diagnosing and resolving issues within applications using the Redux state management library.

These tools provide developers with valuable insights into the application's state, actions, and the flow of data, making the debugging process more efficient and effective.

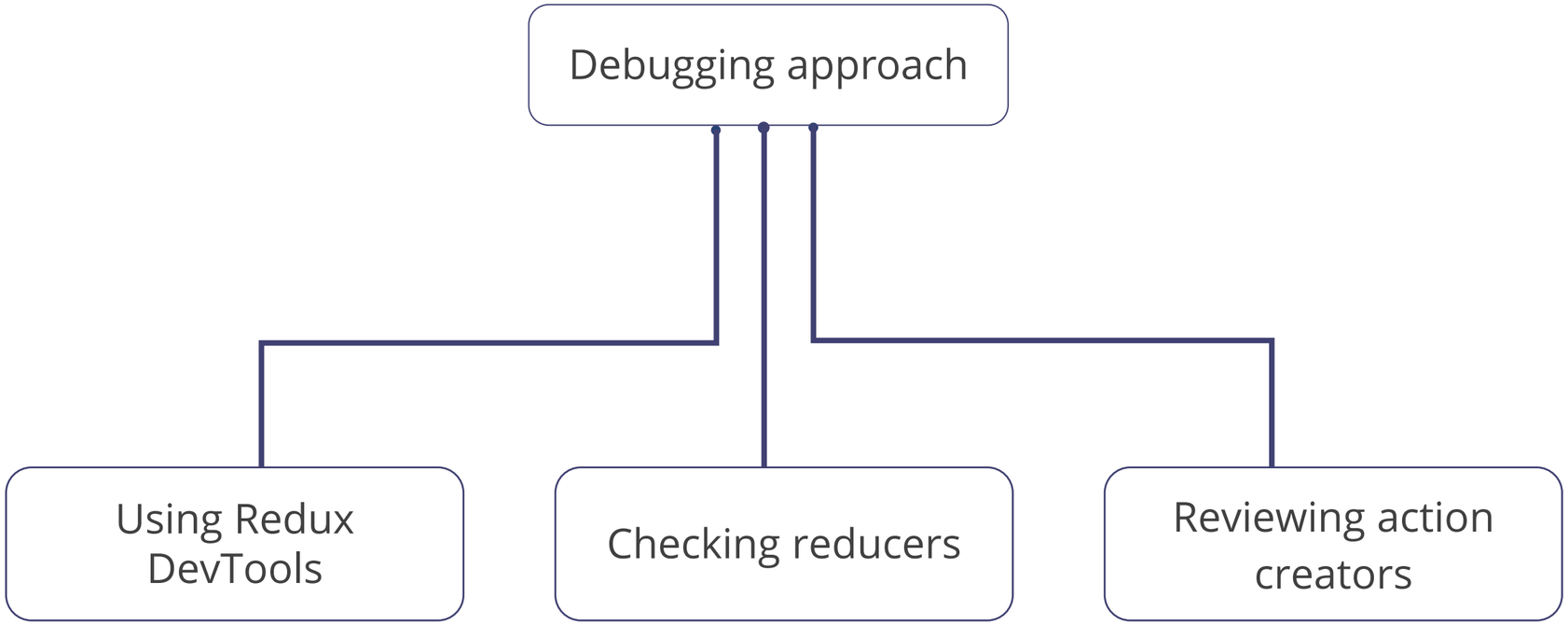
Redux Debugging Tools

The following are the Redux debugging tools:

Redux DevTools extension	It displays a chronological log of dispatched actions, including action types and associated payloads.
Redux logger middleware	It is a middleware that logs actions, previous state, and next state to the console.
React Native debugger	It integrates Redux DevTools, allowing state inspection and action replay within the debugger.

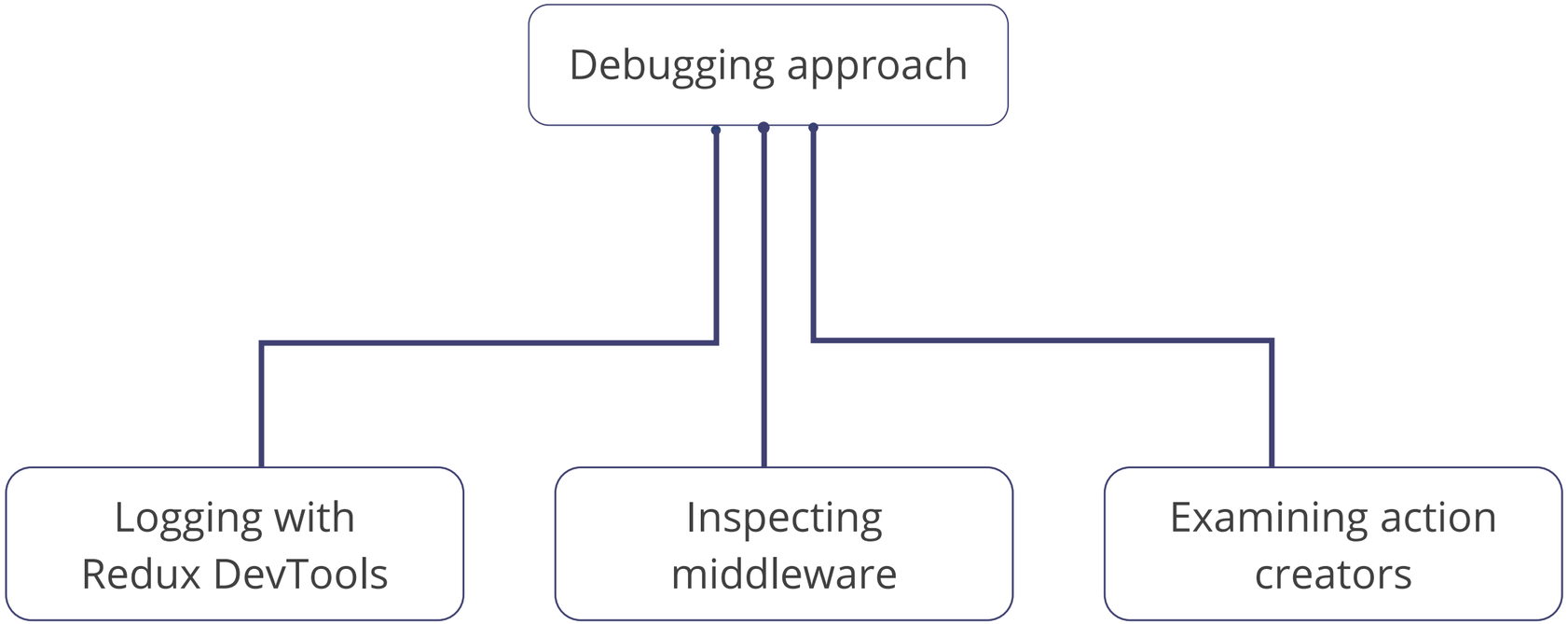
Redux Debugging Scenario: Unexpected State Changes

The application's state is undergoing unexpected modifications, leading to erratic behavior or incorrect rendering.



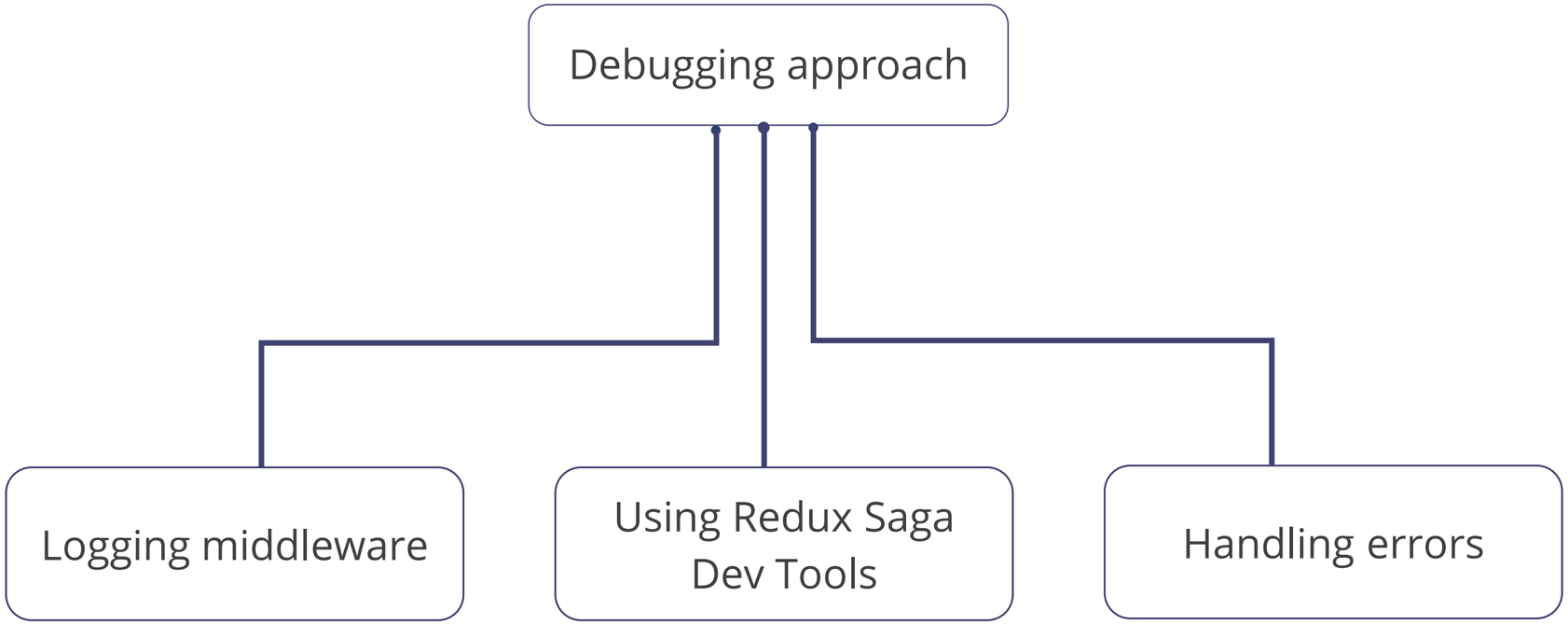
Redux Debugging Scenario: Action Dispatch Issues

Actions are not being dispatched as expected, resulting in unresponsive components or missing state updates.



Redux Debugging Scenario: Asynchronous Operation Challenges

Issues arise when managing asynchronous operations, such as data fetching or API calls, using middleware like Redux Thunk or Redux Saga.



Creating a Simple Debugger React Application

Problem Statement:

Duration: 15 min

You have been assigned a task to create a simple React application that consumes a REST API and debug it with React Developer Tools in the browser using breakpoints.

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up a React project
2. Create user-defined components: GitHubUser.js and Posts.js
3. Add the components to the App.js file
4. Test the application

ASSISTED PRACTICE

Creating an Airport Search React Redux Thunk Application with Debugger

Problem Statement:

Duration: 15 min

You have been assigned a task to create a React application with Redux and Thunk middleware, consuming a REST API to retrieve details based on inputs such as airport name, code, or city name input in the Redux store.

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up the React project
2. Create an airport.json file
3. Create components and services
4. Configure the store and Thunk details in the index.js file
5. Run the application
6. Debug the application using the Redux DevTools with the browser

Introduction to Redux DevTools

Redux DevTools

It provides a graphical user interface that allows developers to gain insights into the inner workings of the Redux applications.

Key features:

Time travel debugging:

This feature allows developers to move backward and forward in the application's state history, significantly aiding in understanding how the state evolves.

Action replay:

This is a powerful capability that enables developers to redispatch actions. It is beneficial for experimenting with different scenarios.

Redux DevTools

Key features:

State inspection:

It provides a detailed view of the application's state at various points in time.

Action log:

It presents a chronological history of dispatched actions.

Diffing state:

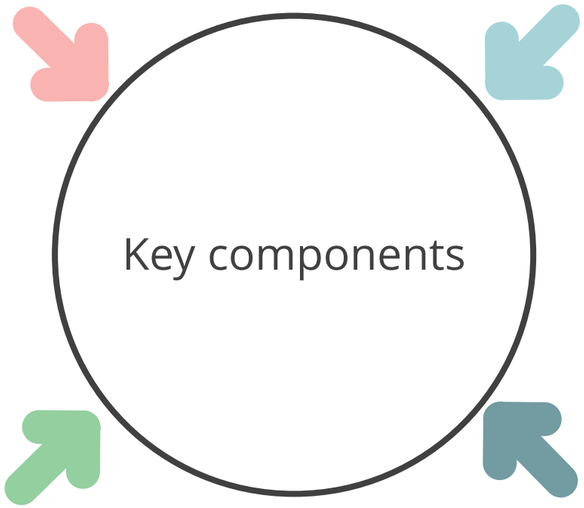
It facilitates the identification of changes in the state caused by specific actions.

Redux DevTools Interface

It is organized into several tabs, each offering unique insights into different aspects of the application.

Inspector:
Displays a detailed view of the current state

State:
Provides a chronological history of the application's state



Actions:
Presents an action log, showing a sequence of dispatched actions

Diff:
Highlights the differences between consecutive states

Best Practices for Using Redux DevTools

Use the time travel feature to revisit different states, analyze changes, and pinpoint when and how issues occurred

Incorporate keyboard shortcuts such as **Ctrl + H** (toggling the DevTools panel) and **Ctrl +]** (moving forward in the action history) to boost productivity

Leverage the test tab to experiment with different action sequences, validate existing behavior, and test newly implemented actions and their interactions

Creating a React Redux Thunk API Employee Operation

Problem Statement:

Duration: 15 min

You have been assigned a task to create the React app with Redux using the Thunk middleware to store and retrieve employee details from a JSON file using Axios.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up the React project
2. Create an employee.json file
3. Create actions, components, and reducers
4. Configure the store and Thunk details in the index.js file
5. Run the application
6. Debug the application using the Redux DevTools with the browser

Key Takeaways

- Debugging involves tracing and fixing unexpected behaviors, ensuring the smooth operation of the Redux-powered application.
- React Developer Tools is a browser extension that is primarily focused on assisting developers in understanding and debugging React components.
- Redux DevTools provides a graphical user interface that allows developers to gain insights into the inner workings of the Redux applications.

Knowledge Check

Knowledge Check 1

Which of the following Redux DevTools features allows developers to move backward and forward in the application's state history?

A. Action replay

B. Time travel debugging

C. State inspection

D. Middleware logging

Knowledge Check 1

Which of the following Redux DevTools features allows developers to move backward and forward in the application's state history?

A. Action replay

B. Time travel debugging

C. State inspection

D. Middleware logging

The correct answer is **B**

Time travel debugging in Redux DevTools enables developers to navigate backward and forward in the application's state history, providing valuable insights into how the state evolves.