Redux

Learning Objectives

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

- Get an insight of state management for handling and maintaining the application state
- Demonstrate the application of creating a React project with Redux, integrating the createStore() function to manage global variables effectively
- Develop a React app that combines two reducers into a single store, ensuring organized and scalable state management
- Explain how the dispatch function is commonly used in conjunction with action creators to encapsulate the process of creating and dispatching actions

Overview of State Management

State refers to the condition or status of an application at a particular point in time.

State management involves handling and maintaining the state of your application, ensuring that different components can access and update it appropriately.

Introduction to Redux

Introduction to Redux

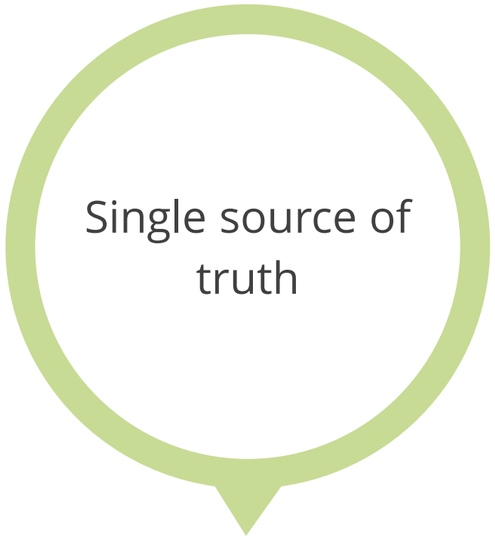
It is a state management library.

Redux

As the complexity of applications increases, efficiently managing the state becomes increasingly challenging. Redux is a predictable state container for JavaScript applications that provides a robust solution to this challenge.

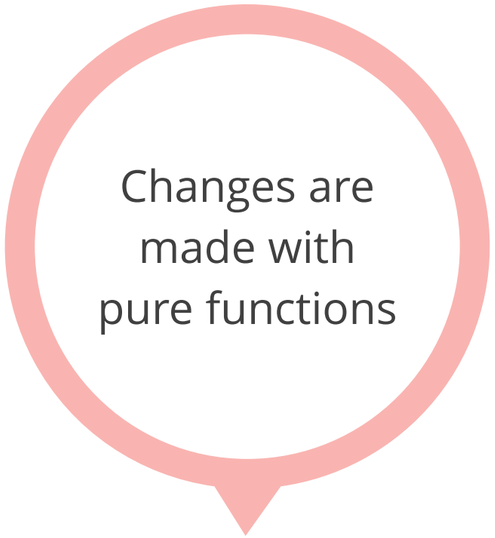
Core Principles of Redux

Redux is based on three fundamental principles:



The entire state of the application is stored in a single JavaScript object called the **store**.

The state is immutable, and its modification can only occur through the emission of an **action**.



Reducers are pure functions that take the current state and an action and return a new state.

Redux Store: The Heart of Redux

About Redux Store

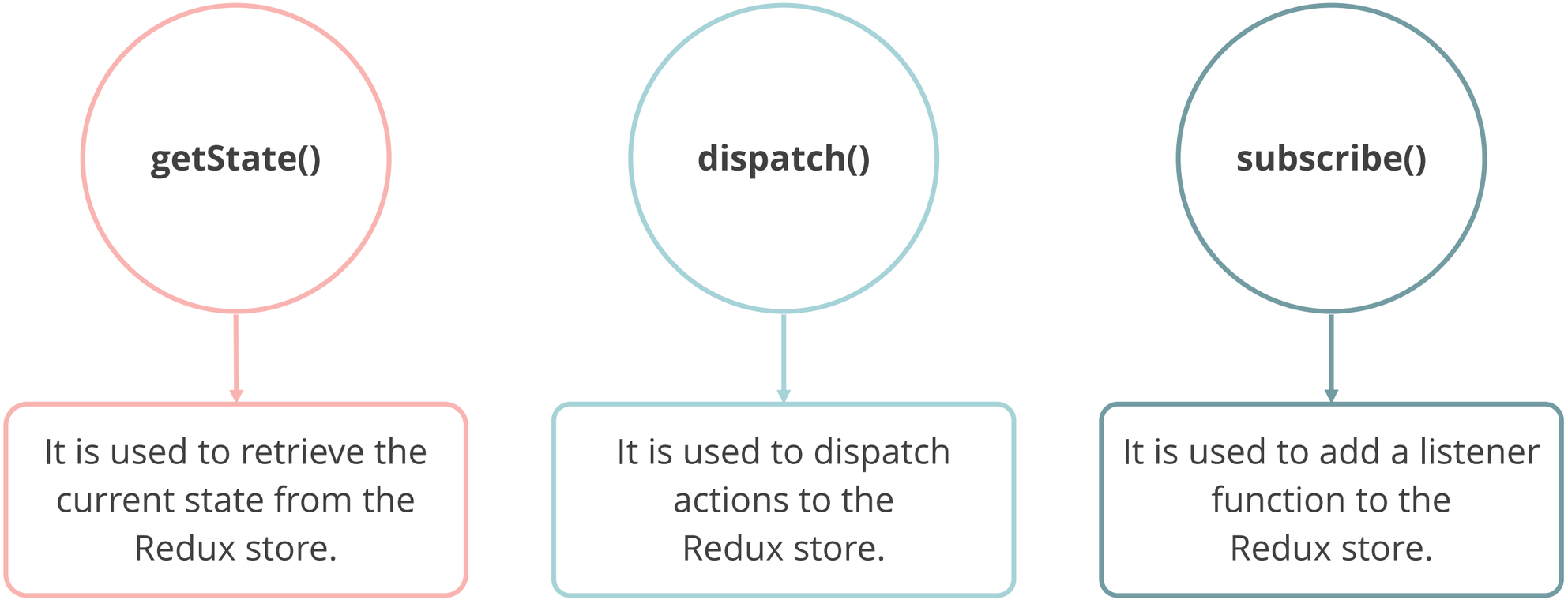
A store holds the whole state tree of the application.

The only way to change the state inside the store is to dispatch an action on it, which triggers the root reducer function to calculate the new state.

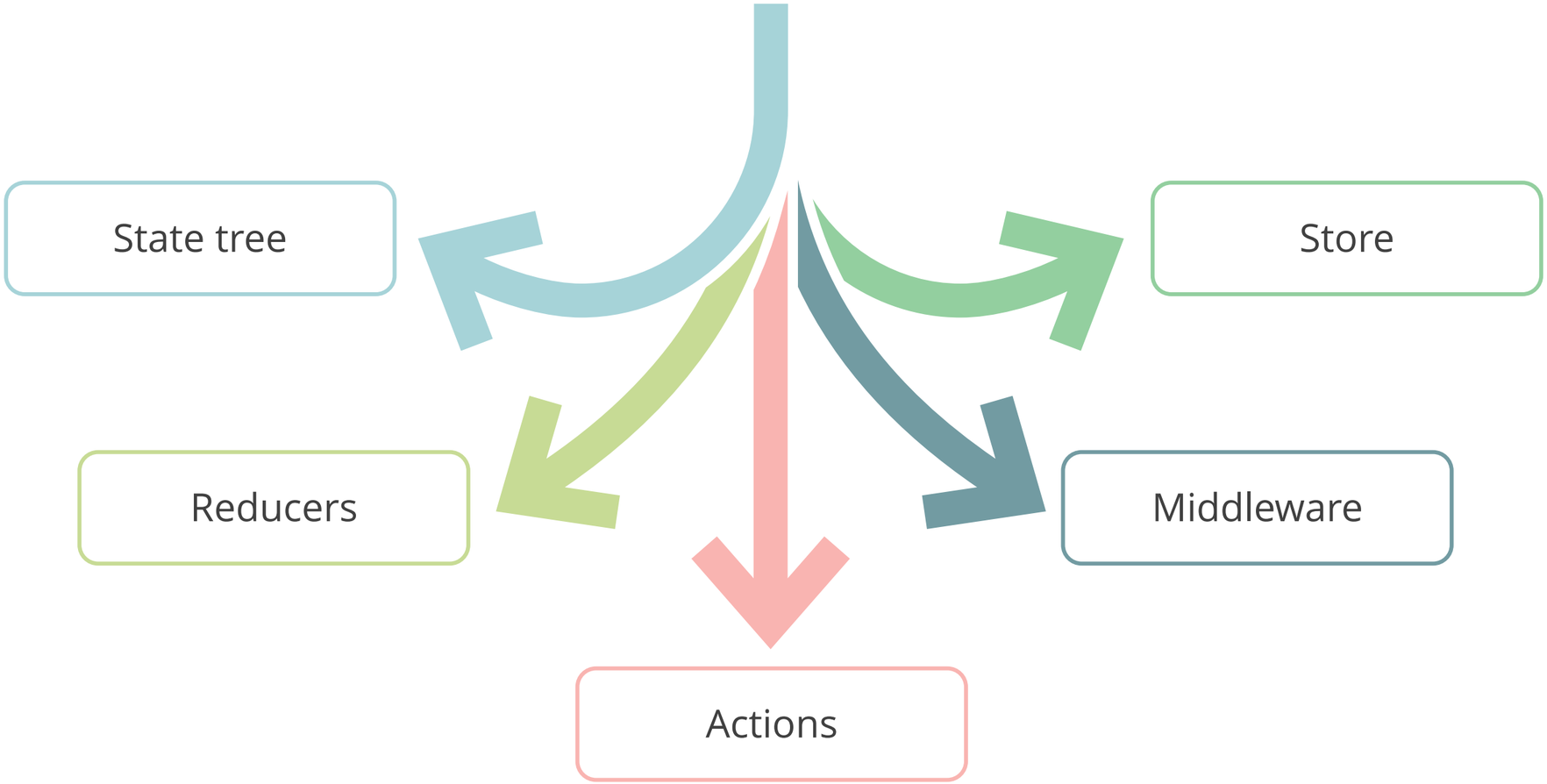
To create a store, one can use the **createStore()** function.

Redux Store: Functions

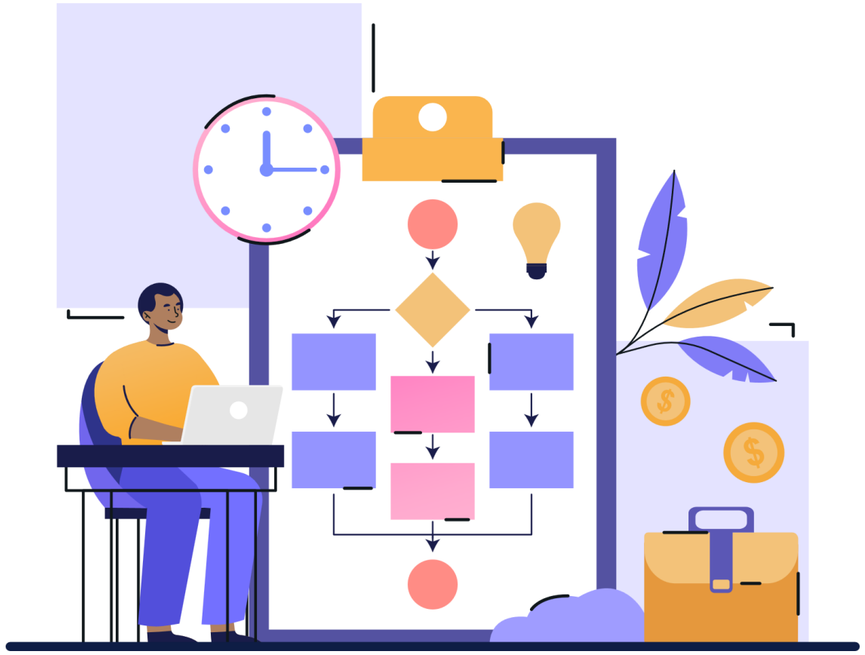
The following functions help to manage the state in a Redux store:



Redux Store: Components



How to Create a Store?



The following are the steps to create a store:

- Install Redux
- Create reducers
- Combine reducers (if necessary)
- Create the store
- Access the store
- Subscribe to changes

Redux Store: Responsibilities

The store handles state modifications in response to dispatched actions.

It ensures that dispatched actions reach the appropriate reducers.

It supports middleware, which allows developers to intercept and augment the dispatch process.

It enforces a predictable flow of data, from actions to reducers and then to the state.

Creating a React Application with Redux to Access the Global Variables

Problem Statement:

Duration: 15 min

You have been assigned the task to create a React application with Redux, showcasing the utilization of a Redux store to manage global variables.

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up the React project
2. Configure the store in the index.js file
3. Create a user-defined component
4. Modify the App.js file

Actions in Redux

Introduction to Actions

They are the core building blocks of the application that describe events or changes.

An action has two key properties:

Type:

It is a string that describes the type of action. It serves as a label indicating what kind of change is being requested.

Example: { type: 'INCREMENT' }

Payload:

It carries additional data relevant to the action. It provides the context or information needed to perform the state modification.

Example:
{ type: 'ADD_TODO', payload: { id: 1, text: 'Learn Redux' } }

Action Types

They refer to the unique identifiers associated with different types of actions that can occur in an application.

Key characteristics:

- Action types are often defined as **string constants**, making them easily recognizable in the code.
- Each action type should be unique within the application, which ensures that actions can be easily identified and processed by the appropriate reducers.
- They are commonly defined in a centralized location, such as a separate file or a dedicated section of a file.
- This centralization promotes maintainability and makes it easier to manage and update action types.

Action Creators

These are functions responsible for creating and returning action objects.

Crafting action creators in Redux involves the following steps:

- Define action type constants
- Create action creators
- Use action creators in components

Reducers in Redux

Reducers in Redux

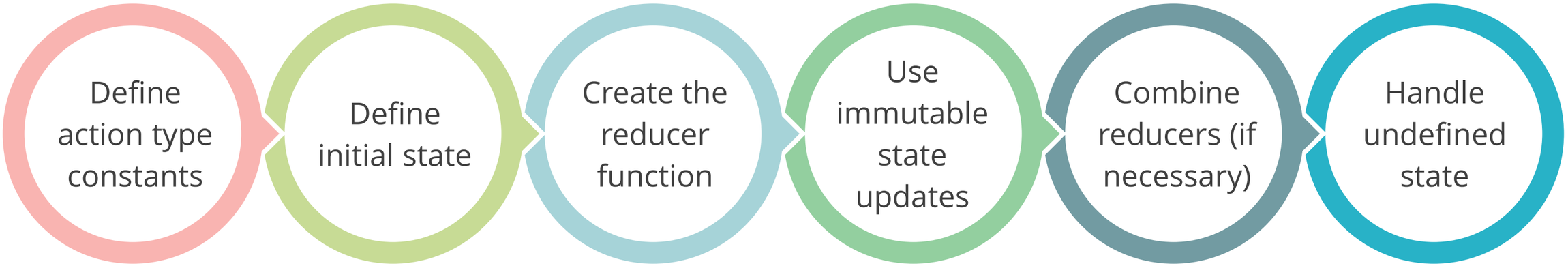
They play a crucial role in managing an application's state by defining alterations triggered by dispatched actions.

Key roles:

- Reducers define the initial state of the application.
- They determine how the state should change in response to dispatched actions.
- They always return a new state object rather than modifying the existing state.
- Many applications feature multiple reducers; the **combineReducers** utility provided by Redux is employed to merge these reducers into a unified root reducer.
- Reducers handle the case where the state is undefined by providing default values.

Designing Reducers

The following are the steps to design a reducer:



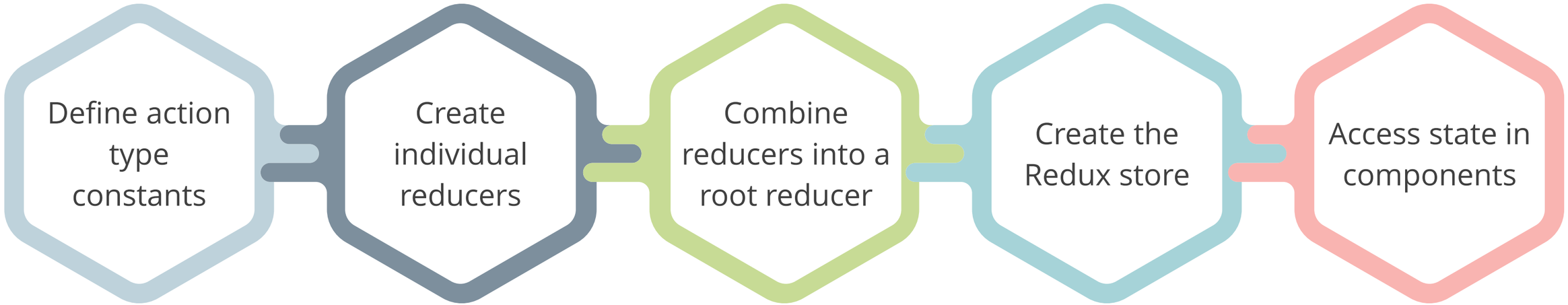
Combining Multiple Reducers

The essential approach for handling various segments of the application state involves combining multiple reducers.

It allows for a modular architecture where different parts of the state are handled independently.

Combining Multiple Reducers

The following are the steps to combine multiple reducers:



Creating a React Application with combineReducers function

Problem Statement:

Duration: 15 min

You have been assigned the task to create a React application with Redux for combining two reducers in one store to perform the task.

ASSISTED PRACTICE

Assisted Practice: Guidelines

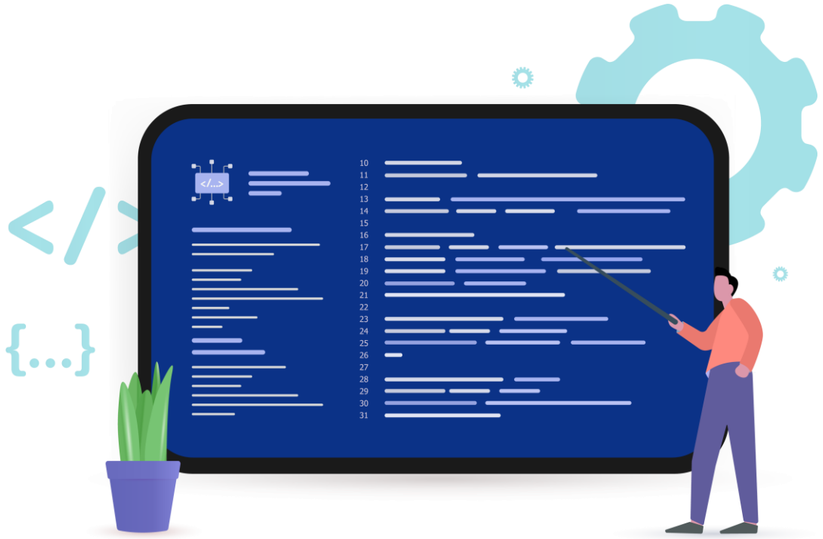
Steps to be followed:

1. Create and set up the React project
2. Create the actions and reducers folder
3. Create an **App.js** and **index.js** file
4. Add Bootstrap features in the index.html file
5. Test the application

Dispatching Actions

Dispatch Function

It is a fundamental function in Redux for initiating state changes by dispatching actions.



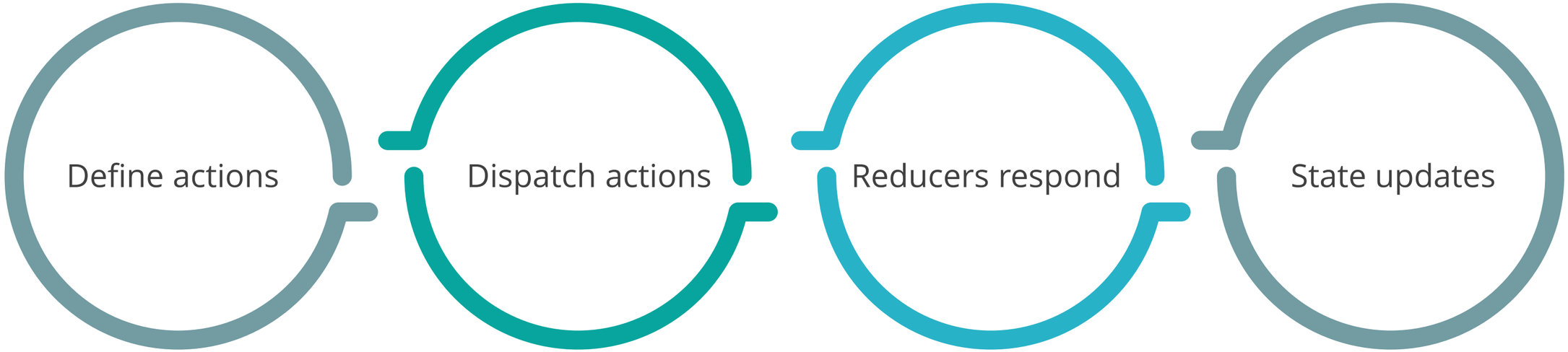
It takes an action object as an argument and triggers the state update process.

The dispatch function is commonly used in conjunction with action creators to encapsulate the process of creating and dispatching actions.

Triggering State Changes

It refers to the process of initiating updates to the application state in response to specific events or user interactions, achieved by dispatching actions.

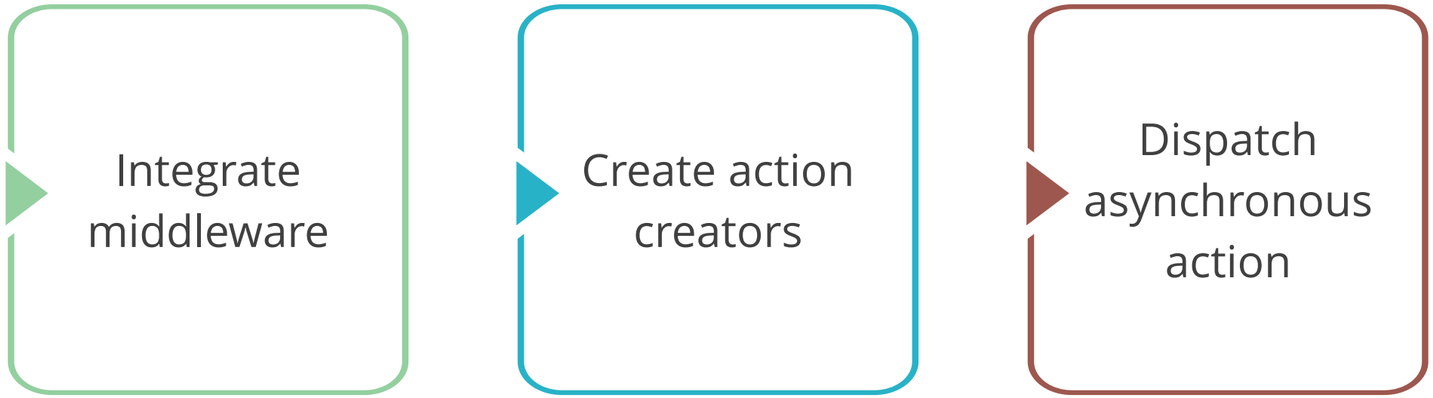
The following are the steps to trigger state changes:



Handling Asynchronous Actions

It refers to the process of dispatching actions that involve asynchronous operations, such as fetching data from an API, and managing the flow of these actions within the Redux architecture.

The following are the steps to handle asynchronous actions:



Creating a React with Redux Counter Application

Problem Statement:

Duration: 15 min

You have been assigned the task to create the React application with Redux to access the counter value in each component and perform the operations on counter variables like increment and decrement.

Assisted Practice: Guidelines

Steps to be followed:

1. Create and set up the React project
2. Configure the Redux
3. Create a user-defined component
4. Test the application

ASSISTED PRACTICE

State Management in Redux Store

Updating the State

It refers to the sequence of steps that occur when the application state changes in response to dispatched actions.

It involves dispatching actions, processing these actions in reducers, and updating the Redux store state accordingly.

Developers can effectively use tools like Redux DevTools to trace actions, inspect state changes, and debug the application with ease.

Immutable State Updates

They involve creating a new state object rather than modifying the existing state.

State immutability ensures that the state is treated as read-only, promoting predictability in how state changes occur.

Immutability aids in debugging by ensuring that each state snapshot is unique.

Debugging State Changes

It involves the process of identifying, inspecting, and resolving issues related to how the application's state is modified over time.

Following are the strategies to enhance debugging state changes:

Utilize the **Redux DevTools** browser extension or integrated tools to perform time-travel debugging and inspect state changes and actions

Implement logging statements strategically within the reducers or middleware to log relevant information about actions and state changes

Creating a React with Redux Employee App

Problem Statement:

Duration: 15 min

You have been assigned the task to create a React application with a Redux store that allows viewing and deleting employee information from the store.

Assisted Practice: Guidelines

Steps to be followed:

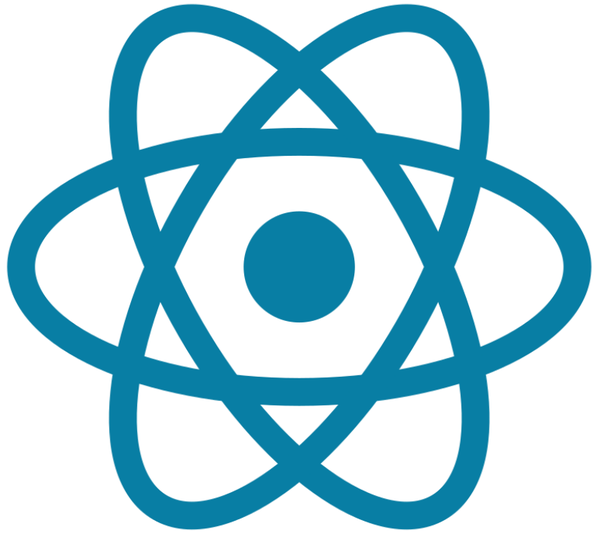
1. Create and set up the React project
2. Configure the Redux
3. Create a user-defined component
4. Test the application

ASSISTED PRACTICE

Connecting Redux with React Components

Overview of the React Redux Library

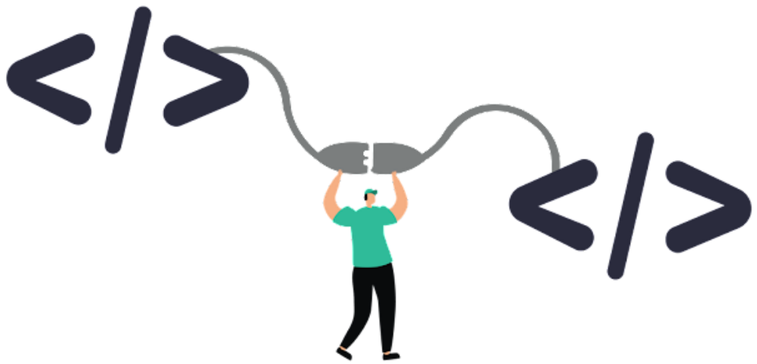
It is an extension of the Redux library that provides React-specific bindings for implementing Redux with React applications.



It automatically optimizes performance, ensuring the component only updates when the relevant data changes.

Connect Function

It is a higher-order function provided by the React Redux library to connect React components to the Redux store.



Steps to integrate the connect function:

- Import the **connect** function
- Define a **mapStateToProps** function
- Connect the components
- Access the Redux state

Mapping State and Dispatch to Props

It is the process of defining functions that map some parts of the Redux state and action dispatchers to the props of a connected React component.

Mapping state to props allows components to access the specific parts of the global state, making relevant data available for rendering.

Mapping dispatch functions to props enables components to dispatch actions to the Redux store, triggering state changes.

Mapping state and dispatch to props facilitates easier testing of components, as these functions can be mocked or overridden during testing.

Key Takeaways

- State management involves handling and maintaining the state of your application, ensuring that different components can access and update it appropriately.
- Actions are the core building blocks that describe events or changes within an application.
- The dispatch function is a fundamental tool in Redux for initiating state changes by dispatching actions.
- The React Redux library is an extension of the Redux library that provides React-specific bindings for implementing Redux with React applications.

Knowledge Check

Knowledge Check

1

What does the *connect* function do?

A. Defines action dispatchers

B. Imports the Redux store

C. Establishes a connection between components and the store

D. Creates a new Redux store

Knowledge Check

1

What does the *connect* function do?

A. Defines action dispatchers

B. Imports the Redux store

C. Establishes a connection between components and the store

D. Creates a new Redux store

The correct answer is **C**

The *connect* function from the React Redux library is used to connect React components to the Redux store.

Knowledge Check

2

How many core principles form the foundation of Redux?

A. Four

B. Two

C. Three

D. Five

Knowledge Check

2

How many core principles form the foundation of Redux?

A. Four

B. Two

C. Three

D. Five

The correct answer is **C**

Redux is based on three core principles: a single source of truth, state is read-only, and changes are made with pure functions.

Knowledge Check

3

What is the purpose of the *payload* property in a Redux action?

A. It defines the type of action.

B. It carries additional data relevant to the action.

C. It specifies the Redux store.

D. It determines the state modification method.

Knowledge Check

3

What is the purpose of the *payload* property in a Redux action?

A. It defines the type of action.

B. It carries additional data relevant to the action.

C. It specifies the Redux store.

D. It determines the state modification method.

The correct answer is **B**

The *payload* property in a Redux action carries additional data that is relevant to the action, providing the context or information needed to perform the state modification.