Vue JS State Management with Pinia

Prerequisites: Working Knowledge of Vue.JS

Day 1: State Management with Pinia

1. Introduction to State Management
   - Importance of State Management in Applications
     - Why state management is crucial
     - Differences between local state and global state
   - Overview of Pinia
     - Introduction to Pinia
     - Comparison with Vuex
     - Pinia Basics

2. State: Understanding the State Object
   - What is state?
   - Creating a state object in Pinia
   - Example: Simple counter state

3. Getters: Accessing and Computing Derived State
   - Defining getters
   - Using getters for derived state
   - Example: Getter to compute double of a counter
4. Mutations: Modifying State
   - How to update state
   - Example: Mutation to increment the counter

5. Actions: Performing Asynchronous Operations and Complex State Changes
   - Role of actions
   - Example: Fetching data from an API and updating state
   - Advanced Pinia Concepts

6. Modules: Organizing State into Modules for Scalability
   - Setting up modules for better scalability
   - Example: User authentication module

7. Integrating Pinia with Vue Router: State Management in Routing
   - Managing state during navigation
   - Example: Storing and accessing route parameters in state

Day 2: Composition API and Composable

1. Introduction to Composition API
   - Differences between Composition API and Options API

- Key differences

2. Why the Composition API was Introduced in Vue 3
   - Motivations and benefits
   - Core Concepts
3. Using Reactive and Ref: Basic Usage and Differences
   - Defining reactive and ref
   - Example: Creating reactive state and refs

4. Lifecycle Hooks in Composition API: Transitioning from Options API to Composition API
   - Lifecycle hooks in Composition API
   - Example: Using mounted and unmounted

5. Reactivity in Depth
   - Reactivity System Overview: How Vue's Reactivity System Works Under the Hood
   - In-depth explanation of Vue's reactivity system

6. Watchers and Computed Properties: Advanced Usage and Best Practices
   - Advanced usage and best practices
   - Example: Using watchers and computed properties

7. Handling Asynchronous Updates: Managing Async Operations in a Reactive Way
   - Managing async updates
   - Example: Fetching data from an API
8. Composable
   - What is a Composable? Understanding the Concept
   - Definition and purpose

9. Composable vs Mixin: Advantages of Composable over Mixins
   - Comparison and advantages
   - Why Use a Composable?
   - Benefits and use cases