### **Phase 1: Deep Dive into Error Handling and Event Handling**

1. **Advanced Error Handling**
   * **Concepts Learned:**
     + Types of JS errors: ReferenceError, TypeError, RangeError, etc.
     + Creating custom errors
     + Graceful degradation & fallback strategies
     + Error propagation
   * **Key Topics:**
     + throw with custom error classes
     + Nested try/catch
     + Centralized error handling patterns
     + Handling async errors in async/await
2. **Event Handling Beyond Basics**

* **Concepts Learned:**
  + Event bubbling, capturing, and delegation
  + Memory leaks with event listeners
  + Passive listeners and performance
* **Key Topics:**
  + event.stopPropagation(), event.preventDefault()
  + Event delegation pattern
  + Removing event listeners
  + Handling large-scale event systems

### **Phase 2: Asynchronous JavaScript (Real-World Usage)**

1. **Mastering Promises and Async Patterns**
   * **Concepts Learned:**
     + Promise chaining and composition
     + Error handling in long chains
     + Promise.all, Promise.any, Promise.allSettled, Promise.race
   * **Key Topics:**
     + Creating utility functions that return promises
     + Handling multiple async tasks with different needs
2. **Async/Await in Practice**

* **Concepts Learned:**
  + Chaining and sequencing async/await operations
  + Error handling with async/await
  + Conditional and concurrent awaits
* **Key Topics:**
  + Refactoring nested promises into async functions
  + Using await in loops correctly

**5. Working with Fetch and Axios**

* **Concepts Learned:**
  + Making HTTP requests to APIs
  + Handling loading, error, and success states
  + CORS, headers, and request config
* **Key Topics:**
  + fetch with async/await + try/catch
  + Axios: configuring base URLs, interceptors
  + Comparison: fetch vs axios

Project break : **Recipe Finder** project

Focus: Making API calls using fetch() with async/await, Implementing UI states: loading spinner, error handling, success rendering, Filtering data in real-time based on user input, Navigating between views using dynamic data, Simulating network delay with Promises for UI testing.

### **Phase 3: Functional Patterns and Closures**

1. **Closures in Real-World Scenarios**
   * **Concepts Learned:**
     + Closures for data privacy and state retention
     + Memory considerations
   * **Key Topics:**
     + Implementing factory functions
     + Closures in event handlers and loops
     + Avoiding common closure pitfalls
2. **Functional Programming with JavaScript**

* **Concepts Learned:**
  + Pure functions, immutability, and composition
  + Currying and partial application
* **Key Topics:**
  + Writing reusable utility functions
  + Using .reduce() to build complex structures
  + Function chaining and pipelines

Project break : **Budget Calculator** project

Focus: Using localStorage for persistent state across sessions, Working with complex data structures (objects, arrays) in the DOM, Dynamic DOM manipulation based on form inputs and calculations, Input validation and real-time UI updates, Prepares mental models for state-based UI—critical for transitioning to React.

### **Phase 4: Advanced ES6+ Techniques and Patterns**

1. **Advanced Module Management**
   * **Concepts Learned:**
     + Dynamic imports
     + Import maps and lazy loading
     + Organizing code into modules effectively
   * **Key Topics:**
     + import() (dynamic)
     + Exporting classes, default/named exports
     + Practical module bundling concepts (Vite/Webpack basics)
2. **JavaScript Patterns and Best Practices**
   * **Concepts Learned:**
     + Module pattern, revealing module pattern
     + Singleton, factory, observer (intro level)
   * **Key Topics:**
     + Writing maintainable code using patterns
     + Encapsulation using IIFE + closure

Project break : **Movie Tracker** project

Focus: Modular architecture using ES6 modules for scalable code organization, Leveraging modern JavaScript features like arrow functions, destructuring, spread/rest operators, and Sets, Implementing dynamic filtering and sorting with efficient array methods, Managing session persistence for user data, Responsive and accessible UI design with event-driven programming, Reinforces real-world patterns and best practices essential for advanced frontend development and preparing for frameworks like React.