Testing Guide
Setting Up and Writing Tests

A Practical Approach to Testing React Applications

Testing Framework Documentation

January 20, 2025

Presentation Overview

Contents

Core Technologies

Primary Testing Tools

- Jest
  - Primary test runner and assertion library
  - Handles test orchestration and reporting
  - Provides mocking capabilities
  - Supports timer mocking for async operations

Core Technologies

Primary Testing Tools

- Jest
  - Primary test runner and assertion library
  - Handles test orchestration and reporting
  - Provides mocking capabilities
  - Supports timer mocking for async operations
- React Testing Library
  - Component testing utility
  - Encourages testing user behavior
  - Provides DOM querying utilities
  - Promotes accessibility-first testing

Contents

Initial Setup

Configuration Files

- jest.config.js - Core Jest configuration
- setupTests.js - Test environment setup

```
// setupTests.js
import `@testing-library/jest-dom';
import { jest } from `@jest/globals';

// Mock timers for all tests
beforeEach(() => {
  jest.useFakeTimers();
});

afterEach(() => {
  jest.runOnlyPendingTimers();
  jest.useRealTimers();
});
```

Directory Structure

Test Organization

Tests are organized following a clear directory structure that mirrors the source code:

```
src/
|-- components/
|   |-- __tests__/         # Component tests
|   `-- ComponentName/
|       `-- __tests__/     # Specific component tests
`-- utils/
    `-- __tests__/         # Utility function tests
```

Contents

Test Structure

Component Test Example

```
describe(`ComponentName', () => {
  describe(`Behavior Category', () => {
    test(`should exhibit specific behavior', () => {
      // Test implementation
    });
  });
});
```

Key Elements

- Clear test grouping using describe blocks
- Focused, single-responsibility tests
- Proper setup and teardown
- Consistent naming conventions

Mock Management

```
// Mock dependencies
jest.mock(`../dependency', () => ({
  someFunction: jest.fn()
}));

// Helper setup
const setupComponent = (props = {}) => {
  render(<Component {...props} />);
};
```

Best Practices

- Mock external dependencies
- Create reusable setup functions
- Clean state between tests
- Control timer management

Contents

Types of Tests

1. View-Based Tests
    - Container tests (component composition)
    - Mobile view specific tests
    - Desktop view specific tests
    - View-specific interactions and layouts

Types of Tests

1. View-Based Tests
   - Container tests (component composition)
   - Mobile view specific tests
   - Desktop view specific tests
   - View-specific interactions and layouts
2. Shared Component Tests
   - Reusable component behavior
   - Common functionality
   - Shared hooks and utilities

Types of Tests

1. View-Based Tests
   - Container tests (component composition)
   - Mobile view specific tests
   - Desktop view specific tests
   - View-specific interactions and layouts
2. Shared Component Tests
   - Reusable component behavior
   - Common functionality
   - Shared hooks and utilities
3. Integration Tests
   - Component interactions
   - State management
   - Data flow

Contents

Testing Guidelines

Core Principles

- Test behavior over implementation
- One assertion per test
- Clear test descriptions
- Proper test isolation

Common Pitfalls to Avoid

- Testing implementation details
- Brittle selectors
- Complex test setups
- Shared state between tests

Test-Driven Development

1. Write test first
   - Define expected behavior
   - Create failing test

Test-Driven Development

1. Write test first
   - Define expected behavior
   - Create failing test
2. Implement minimal code
   - Write just enough code to pass
   - Focus on functionality

Test-Driven Development

1. Write test first
   - Define expected behavior
   - Create failing test
2. Implement minimal code
   - Write just enough code to pass
   - Focus on functionality
3. Refactor
   - Clean up implementation
   - Maintain test coverage

Contents

Keeping Tests Healthy

Regular Reviews

- Test coverage analysis
- Performance monitoring
- Pattern consistency
- Documentation updates

Updates

- Framework version updates
- Dependency management
- Best practices alignment
- Test suite optimization

Contents

Continuous Enhancement

Areas for Growth

- Coverage expansion

Continuous Enhancement

Areas for Growth

- Coverage expansion
- Performance optimization

Continuous Enhancement

Areas for Growth

- Coverage expansion
- Performance optimization
- Documentation improvements

Continuous Enhancement

Areas for Growth

- Coverage expansion
- Performance optimization
- Documentation improvements
- Testing patterns library