Unit Testing ReactJS Apps (Jest & Enzyme)

Q1. What is the main purpose of Jest in React projects?

1. State management
2. **Unit testing framework**
3. CSS styling tool
4. Routing library

Ans: **b**

Q2. Which command is commonly used to run Jest tests?

1. npm run start
2. **npm run test**
3. npm run build
4. npm run lint

Ans: **b**

Q3. Enzyme is primarily used for:

1. API requests
2. **Component testing**
3. Redux store management
4. Internationalization

Ans: **b**

Q4. Which Jest function is used to group related test cases?

1. test()
2. expect()
3. **describe()**
4. it()

Ans: **c**

Q5. In Jest, which function is used to make assertions?

1. test()
2. **expect()**
3. render()
4. shallow()

Ans: **b**

Q6. What does the shallow() method from Enzyme do?

1. Fully renders the component tree

b) **Renders only the given component without its children**

c)Tests API endpoints

d) Creates Redux store mocks Ans: **b**

Q7. Which function is used to render a component with full DOM in Enzyme?

1. shallow()
2. render()
3. **mount()**
4. snapshot()

Ans: **c**

Q8. Which Jest matcher is used to check strict equality?

1. **toBe()**
2. toEqual()
3. toMatch()
4. toContain() Answer: a

Q9. What does Jest's toEqual() check for?

1. Reference equality
2. **Deep equality of objects/arrays**
3. String matching
4. Boolean comparison

Ans: **b**

Q10. Which Enzyme method simulates user interactions?

1. **simulate()**
2. trigger()
3. act()
4. mockEvent()

Ans: **a**

Q11. Jest's beforeEach() function is used to:

1. Run once before all tests
2. **Run before every test case**
3. Run once after all tests
4. Run only if tests fail

Ans: **b**

Q12. Snapshot testing in Jest ensures that:

1. API responses match schema
2. **Component output hasn’t unexpectedly changed**
3. Redux store is updated
4. Database schema is consistent

Ans: **b**

Q13. Which file extension is commonly used for Jest test files?

1. .jsx
2. **.test.js**
3. .snap.js
4. .mock.js

Ans: **b**

Q14. Which Jest function is used to mock dependencies?

1. jest.fn()
2. jest.mock()
3. jest.spyOn()
4. **All of the above**

Ans: **d**

Q15. What does enzyme-adapter-react-16 (or latest) do?

1. Adds Redux support
2. **Bridges Enzyme with React version**
3. Enables Jest matchers
4. Creates snapshots

Ans: **b**

Q16. In Jest, which function is used to run a single test file?

1. **npm test fileName.test.js**
2. jest --runSingle
3. npm run build fileName
4. testOnly fileName

Answer: **a**

Q17. Enzyme’s find() method is used to:

1. **Locate a DOM node or component**
2. Simulate user clicks
3. Create a snapshot
4. Mount the component

Ans: **a**

Q18. Which Jest configuration file is commonly used?

1. **jest.config.json**
2. jestfile.json
3. test.config.js
4. enzyme.config.js

Ans: **a**

Q19. Which Jest function is used to check if a function is called?

1. toBeCalled()
2. **toHaveBeenCalled()**
3. wasCalled()
4. fnCalled()

Ans: **b**

Q20. In Jest, which option enables coverage reports?

1. npm test --verbose
2. **npm test --coverage**
3. npm test --watch
4. npm test --report

Ans: **b**

# E2E Testing using Cypress

**Q21.** Cypress is mainly used for:

1. Backend testing
2. API testing
3. **End-to-End testing**
4. Database testing

Ans: **c**

**Q22.** Cypress tests run directly inside the:

1. **Browser**
2. Server
3. Database
4. IDE

**Ans: a**

**Q23.** Which Cypress command is used to visit a page?

1. cy.load()
2. cy.goto()
3. **cy.visit()**
4. cy.open()

**Ans: c**

**Q24.** Cypress test files are usually written in:

1. HTML
2. **JavaScript/TypeScript**
3. Python
4. C#

**Ans: b**

**Q25.** Cypress default folder for integration tests is:

1. /cypress/tests/
2. **/cypress/integration/**
3. /tests/integration/
4. /src/tests/

**Ans: b**

**Q26.** Cypress supports which type of testing?

1. Unit
2. Component
3. Integration
4. **All of the above**

**Ans: d**

**Q27.** To get an element by ID in Cypress:

1. cy.getElement('#id')
2. cy.find('#id')
3. **cy.get('#id')**
4. cy.locate('#id')

**Ans: c**

**Q28.** Which Cypress command is used to simulate a click?

1. cy.press()
2. cy.tap()
3. **cy.click()**
4. cy.triggerClick()

**Ans: c**

**Q29.** Cypress provides:

1. **Real browser environment**
2. Virtual DOM only
3. API mocks only
4. Only CLI support

**Ans: a**

**Q30.** Cypress runs asynchronously but provides:

1. Callbacks
2. **Automatic waiting**
3. Promises only
4. Manual waits

**Ans: b**

**Q31.** Cypress command to type text into an input:

1. cy.enter('text')
2. **cy.type('text')**
3. cy.write('text')
4. cy.input('text')

**Ans: b**

**Q32.** Cypress command to assert visibility:

1. cy.get().visible()
2. **cy.should('be.visible')**
3. cy.expect('visible')
4. cy.assert('visible')

**Ans: b**

**Q33.** Cypress supports test retries with:

1. Retry block
2. **Cypress.config({ retries })**
3. cy.retry()
4. cy.loop()

**Ans: b**

**Q34.** Cypress dashboard is used for:

1. **Real-time test results & analytics**
2. Code compilation
3. Storing test code
4. Database queries

**Ans: a**

**Q35.** Cypress automatically handles:

1. **Async code & waits**
2. SQL queries
3. Cloud deployments
4. Webpack builds

**Ans: a**

Redux Data, Async and Data Fetching

Q36. In Redux, the state must always be treated as:

1. Mutable
2. **Immutable**
3. Temporary
4. None of the above

Ans: **b**

Q37. Which function in Redux is used to combine multiple reducers into one?

1. createReducer
2. applyMiddleware
3. **combineReducers**
4. rootReducer

Ans: **c**

Q38. Which Redux middleware is commonly used for handling asynchronous operations?

1. redux-logger
2. **redux-thunk**
3. redux-toolkit
4. redux-persist

Ans: **b**

Q39. In Redux, what does dispatch() do?

1. Updates the reducer directly
2. **Sends an action to the store**
3. Returns the current state
4. Initializes the store

Ans: **b**

Q40. What is the correct order of Redux data flow?

1. Store → Reducer → Action → UI
2. **UI → Action → Reducer → Store → UI**
3. Reducer → Store → Action → UI
4. Action → UI → Reducer → Store

Ans: **b**

Q41. When fetching data in Redux, where should the API call usually be placed?

1. Inside the reducer
2. **Inside the action creator (with middleware)**
3. Inside the store directly
4. Inside the component only

Ans: **b**

Q42. Which hook is often used in React-Redux for accessing state from the store?

1. useEffect
2. useReducer
3. **useSelector**
4. useContext

Ans: **c**

Q43. Which hook is used in React-Redux to dispatch actions?

1. **useDispatch**
2. useSelector
3. useReducer
4. useAction

Ans: **a**

Q44. What does an action in Redux contain?

1. Reducers and state
2. State and middleware
3. **Type and payload**
4. Store and UI

Ans: **c**

Q45. What happens if you try to mutate Redux state directly?

1. Redux automatically corrects it
2. State updates without problems
3. **It breaks time-travel debugging and immutability principles**
4. Nothing happens

Ans: **c**

Q46. Which of the following is NOT true about Redux store?

1. It holds the application state
2. **It allows direct modification of state**
3. It is updated only through dispatching actions
4. It is created using createStore or configureStore

Ans: **b**

Q47. What is the main purpose of Redux middleware?

1. To directly update the reducer
2. To log only errors
3. **To intercept actions before they reach the reducer**
4. To remove boilerplate code

Ans: **c**

Q48. Which library is recommended by the Redux team for writing Redux logic?

1. redux-observables
2. **redux-toolkit**
3. redux-thunk
4. redux-persist

Ans: **b**

Q49. What does the mapStateToProps function do in React-Redux?

1. Maps component props to reducer
2. **Maps Redux state to React component props**
3. Maps React component props to Redux store
4. Maps dispatch to reducers

Ans: **b**

Q50. If you want to initialize state in Redux with server data, which lifecycle stage is best?

1. **Before rendering the component (useEffect)**
2. Inside the reducer directly
3. Only during store creation
4. Inside UI event handlers

Ans: **a**

# RxJS & Redux-Observables, Reducers & Actions

**Q51.** What does RxJS primarily help with in React applications?

1. Managing UI components
2. **Handling asynchronous data streams**
3. Styling components
4. Building server-side apps

**Ans: b**

**Q52.** What is a Redux-Observable?

1. **A middleware for handling async logic with RxJS**
2. A library for styling React apps
3. A testing framework for React
4. A server-side rendering library

**Ans: a**

**Q53.** Which RxJS operator is commonly used in Redux-Observable epics for mapping actions?

1. **switchMap**
2. mapState
3. reduce
4. dispatchMap

**Ans: a**

**Q54.** In Redux, what is the role of a reducer?

1. Dispatching actions
2. **Describing state changes based on actions**
3. Fetching data from API
4. Handling authentication

**Ans: b**

**Q55.** Which of the following best describes Redux actions?

1. Functions that modify state directly
2. Pure functions
3. **Plain JavaScript objects describing changes**
4. Observables that emit state

**Ans: b**

**Q56.** Which operator cancels the previous observable when a new one is emitted?

1. mergeMap
2. **switchMap**
3. concatMap
4. map

**Ans: b**

**Q57.** What is the purpose of implementing Undo History in Redux?

1. **To support rollback of UI state**
2. To manage API requests
3. To improve performance
4. To simplify reducers

**Ans: a**

**Q58.** Which Redux concept is most important for supporting undo/redo?

1. Middleware
2. Reducer composition
3. **Immutable state**
4. Async actions

**Ans: c**

**Q59.** What does ImmutableJS provide?

1. Fast rendering
2. **Persistent, immutable data structures**
3. Authentication management
4. A testing framework

**Ans: b**

**Q60.** Which of the following is a key advantage of ImmutableJS in Redux apps?

1. Larger bundle size
2. Direct state mutation
3. **Performance optimization with structural sharing**
4. Automatic API fetching

**Ans: c**

**Q61.** In RxJS, which operator is best for handling multiple values sequentially without cancellation?

1. switchMap
2. **concatMap**
3. mergeMap
4. flatMap

**Ans: b**

**Q62.** Which function is used in Redux to combine multiple reducers?

1. mergeReducers()
2. joinReducers()
3. **combineReducers**()
4. useReducers()

**Ans: c**

**Q63.** In Redux-Observable, what is an Epic?

1. A reducer that handles async data
2. **A middleware function that listens for actions and returns new actions**
3. A UI component that renders state
4. A Redux store enhancer

**Ans: b**

**Q64.** ImmutableJS provides which method to update deeply nested data without mutation?

1. setDeep()
2. **updateIn()**
3. pushState()
4. changeIn()

**Ans: b**

**Q65.** In Undo History implementation, which Redux principle ensures previous states can be restored?

1. Single source of truth
2. Actions must be pure
3. **State is immutable**
4. Store is asynchronous

**Ans: c**

Redux-Thunk & Redux-Saga MCQs

1. What is the primary purpose of Redux-Thunk?
   1. To handle routing in React apps
   2. **To write action creators that return functions instead of actions**
   3. To test Redux reducers
   4. To optimize React rendering

Ans: **b**

1. In Redux-Thunk, the function returned by an action creator receives:
   1. Only dispatch
   2. Only getState
   3. **Both dispatch and getState**
   4. Neither

Ans: **c**

1. Which of the following is a use case for Redux-Thunk?
   1. Testing React components
   2. **Managing asynchronous API calls**
   3. Writing reducers
   4. Handling routing

Ans: **b**

1. Redux-Saga is based on which JavaScript feature?
   1. Promises
   2. **Generators**
   3. Callbacks
   4. Async/Await

Ans: **b**

1. In Redux-Saga, which effect is used to call asynchronous functions?
   1. put
   2. **call**
   3. take
   4. delay

Ans: **b**

1. Which Redux middleware is best for complex async workflows like cancellation and sequencing?
   1. Redux-Thunk
   2. **Redux-Saga**
   3. Redux-Logger
   4. Redux-Observable

Ans: **b**

1. The put effect in Redux-Saga is used to:
   1. **Dispatch an action**
   2. Call an API
   3. Cancel a task
   4. Create a reducer

Ans: **a**

1. In Redux-Saga, the takeEvery effect does what?
   1. Cancels previous sagas
   2. **Runs a saga for every matched action**
   3. Runs only the latest action saga
   4. Runs sagas sequentially

Ans: **b**

1. Which effect in Redux-Saga runs only the latest task and cancels previous ones?
   1. **takeLatest**
   2. takeEvery
   3. fork
   4. join

Ans : **a**

1. What is a key difference between Redux-Thunk and Redux-Saga?
   1. **Thunk uses Promises, Saga uses Generators**
   2. Thunk is synchronous, Saga is asynchronous
   3. Thunk is faster than Saga
   4. Saga does not support async

Ans: **a**

1. Which middleware allows you to retry failed API calls automatically?
   1. Redux-Thunk
   2. **Redux-Saga**
   3. Redux-Persist
   4. Redux-Logger

Ans: **b**

1. In Redux-Saga, yield call(apiFunction) ensures:
   1. **The function is executed asynchronously**
   2. The action is dispatched
   3. The reducer is updated
   4. The store is replaced

Ans: **a**

1. Which of these is NOT true about Redux-Thunk?
   1. It is simpler to learn than Redux-Saga
   2. It uses functions instead of plain actions
   3. It is good for small to medium apps
   4. **It requires generator functions**

Ans: **d**

1. Which effect is used to pause execution in Redux-Saga for a given time?
   1. wait
   2. **delay**
   3. timeout
   4. sleep

Ans: **b**

1. Which scenario would benefit more from Redux-Saga than Redux-Thunk?
   1. Simple API calls
   2. **Complex async workflows with cancellation**
   3. Dispatching synchronous actions
   4. Static state

Ans: **b**

1. What is the main purpose of Redux-Thunk?
   1. **Handle asynchronous logic in Redux**
   2. Manage routing in Redux apps
   3. Optimize rendering performance
   4. Create reducers automatically

Ans: **a**

1. Redux-Thunk allows dispatching of:
   1. Only objects
   2. **Functions and objects**
   3. Only strings
   4. Reducers directly

Ans: **b**

1. Which middleware is required to enable Redux-Thunk?
   1. redux-saga
   2. redux-devtools
   3. **redux-thunk**
   4. redux-persist

Ans: **c**

1. In Redux-Saga, which effect is used to call an asynchronous function?
   1. take
   2. **call**
   3. put
   4. select

Ans: **b**

1. What does the put effect in Redux-Saga do?
   1. Calls an API
   2. **Dispatches an action**
   3. Waits for an action
   4. Cancels a task

Ans: **b**

1. Redux-Saga is built on top of:
   1. Async/Await
   2. Observables
   3. **Generator functions**
   4. Promises only

Ans: **c**

1. In Redux-Saga, the takeLatest effect is used to:
   1. Run all actions in sequence
   2. **Cancel previous tasks and run only the latest one**
   3. Queue all actions without canceling
   4. Run actions in parallel

Ans: **b**

1. Redux-Thunk is best suited for:
   1. Complex async flows with cancellation
   2. **Simple async logic like API calls**
   3. State immutability handling
   4. Reducer composition

Ans: **b**

1. Which effect in Redux-Saga is used to watch for dispatched actions?
   1. **take**
   2. call
   3. race
   4. put

Ans: **a**

1. Redux-Saga can handle:
   1. Only synchronous logic
   2. **Complex asynchronous workflows**
   3. CSS styling
   4. DOM rendering

Ans: **b**

1. Which of the following is TRUE about Redux-Thunk?
   1. It uses generator functions
   2. **It allows writing action creators that return functions**
   3. It replaces reducers
   4. It is built on RxJS

Ans: **b**

1. In Redux-Saga, the select effect is used to:
   1. Fetch data from an API
   2. **Access the Redux store state**
   3. Dispatch an action
   4. Cancel an effect

Ans: **b**

1. Which is an advantage of Redux-Saga over Redux-Thunk?
   1. Simpler for beginners
   2. **Handles side effects more declaratively**
   3. No need for middleware
   4. Faster rendering always

Ans: **b**

1. Which effect in Redux-Saga allows running multiple effects in parallel?
   1. race
   2. **all**
   3. call
   4. fork

Ans: **b**

1. In Redux-Saga, what does the fork effect do?
   1. Run a blocking call
   2. **Spawn a non-blocking task**
   3. Cancel all tasks
   4. Dispatch a reducer

Ans: **b**

React i18n (15 MCQs)

Q96. What does i18n stand for in software development?

1. Integration
2. **Internationalization**
3. Interaction
4. Initialization

Ans:  **b**

Q97. How many letters are between the first and last letters in “Internationalization” that form i18n?

1. 15
2. 17
3. **18**
4. 19

Ans: **c**

Q98. Which React library is most commonly used for i18n support?

1. react-intl
2. react-localize-redux
3. **react-i18next**
4. next-translate

Ans: **c**

Q99. In react-i18next, what is the hook used to access translation functions?

1. useLocale
2. useLang
3. useI18n
4. **useTranslation**

Ans: **d**

Q100. Which JSON structure is typically used for storing translations?

1. Array of strings
2. **Key-value pairs**
3. CSV format
4. XML format

Ans: **b**

Q101. In i18n, what is “l10n”?

1. **Localization**
2. Lexicalization
3. Legalization
4. Language-switching

Ans: **a**

Q102. Which of the following is NOT an i18n challenge?

1. Different date formats
2. Currency conversion
3. String interpolation
4. **Component state management**

Ans: **d**

Q103. In i18next, what option allows you to fallback to a default language?

1. defaultLang
2. **fallbackLng**
3. baseLocale
4. backupLang

Ans: **b**

Q104. Which of these locales is valid for US English?

1. en-uk
2. **en-us**
3. eng-us
4. en-english

Ans: **b**

Q105. What is the purpose of ICU message formatting in i18n?

1. To format database queries
2. **To support pluralization and gender rules**
3. To compress translation files
4. To sort translation keys

Ans: **b**

Q106. What does the Trans component in react-i18next help with?

1. Language detection
2. **Rendering translated strings with React elements inside**
3. Switching between locales
4. Storing translations in cookies

Ans: **b**

Q107. Which of the following is NOT a feature of react-i18next?

1. Lazy loading translations
2. Context-based translations
3. Automatic pluralization
4. **Automatic code-splitting**

Ans: **d**

Q108. If a translation key is missing, what will i18next usually display?

1. Error message
2. Empty string
3. **The key itself**
4. Default English

Ans: **c**

Q109. Which React feature can help with dynamic text changes in multiple languages?

1. **Context API**
2. useMemo
3. useEffect
4. Suspense

Ans: **a**

Q110. Which statement is TRUE about localization?

1. It always uses English as fallback
2. **It adapts content to specific regions/cultures**
3. It only changes the text language
4. It ignores formatting of dates and numbers

Ans: **b**