

Advanced Locators & Synchronization

Multiple Choice Question 1

Which locator strategy provides high resilience when element IDs or classes frequently change in a web application?

Options:

- A. Absolute XPath
- B. CSS selector using wildcard attribute match
- C. Link text
- D. Tag name

Skill: Advanced Locators

Subskill: Robust XPath and CSS Strategies

Competency: Apply flexible locator methods for dynamic elements

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
 - B. CSS attribute wildcard matching handles dynamic attributes efficiently.
 - Incorrect Options Feedback:
 - o A. Absolute paths are brittle.
 - o C. Link text only applies to anchors.
 - o D. Tag names are too generic.
-

Multiple Choice Question 2

What major problem do synchronization-related test failures commonly indicate?

Options:

- A. Incorrect driver setup
- B. Missing configuration files
- C. Timing mismatch between scripts and application behavior
- D. Data type mismatch

Skill: Synchronization

Subskill: Wait Mechanisms

Competency: Identify causes of synchronization issues

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - C. Failing synchronization denotes timing gaps between element load and test execution.
- Incorrect Options Feedback:
 - o A. Driver issues are unrelated.
 - o B. Config files affect setup, not synchronization.
 - o D. Data types are language-level issues.

Multiple Choice Question 3

When should synchronization techniques be optimized in test execution?

Options:

- A. During code compilation
- B. After encountering “Element not interactable” errors
- C. While setting the driver path
- D. During assertion evaluation

Skill: Synchronization Strategy

Subskill: Practical Debugging

Competency: Diagnose synchronization needs

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
 - B. Such errors imply page elements are not ready for interaction, prompting synchronization.
- Incorrect Options Feedback:
 - o A. Compilation does not test runtime behavior.
 - o C. Driver path setup is static.
 - o D. Assertions check logic outcomes, not timing.

Multiple Choice Question 4

Which approach ensures the test waits only as long as needed for a condition to become true?

Options:

- A. Implicit Wait
- B. Explicit Wait
- C. Thread.sleep()
- D. Static Timeout

Skill: Synchronization

Subskill: Efficient Waiting

Competency: Apply conditional waiting

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. Explicit Wait terminates immediately once conditions are met, avoiding idle time.
 - Incorrect Options Feedback:
 - o A. Implicit Wait operates globally.
 - o C. Thread.sleep() is fixed-delay based.
 - o D. Static timeouts waste resources.
-

Multiple Choice Question 5

What best practice minimizes synchronization failures in large regressions?

Options:

- A. Increasing sleep duration
- B. Applying mixed waits based on UI load performance
- C. Disabling waits
- D. Relying only on implicit wait

Skill: Synchronization

Subskill: Framework Optimization

Competency: Achieve reliable synchronization design

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - B. Using tailored combinations of waits ensures stability without slowing tests excessively.
 - Incorrect Options Feedback:
 - o A. Manual sleep leads to inefficiency.
 - o C. Disabling waits reduces reliability.
 - o D. Implicit wait alone lacks fine control.
-

Advanced XPath: contains()

Multiple Choice Question 1

Which statement accurately describes the role of `contains()` in XPath expressions?

Options:

- A. Matches elements with exact attribute values
- B. Matches elements with partial text or attribute values
- C. Matches by element tag only
- D. Restricts search to root-level nodes

Skill: XPath

Subskill: Partial Matching

Competency: Use dynamic attribute-based selection

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. `contains()` enables matching partial attribute or text strings, essential for dynamic UIs.
- Incorrect Options Feedback:
 - o A. Exact matching uses `=`.
 - o C. `contains()` is not tag-specific.
 - o D. It applies at any DOM level.

Multiple Choice Question 2

Which XPath correctly finds elements whose attribute “id” contains the substring “user”?

Options:

- A. `//*[@id='user']`
- B. `//*[contains(@id, 'user')]`
- C. `//*[starts-with(@id, 'user')]`
- D. `//*[ends-with(@id, 'user')]`

Skill: XPath

Subskill: Attribute Filtering

Competency: Build adaptive locators

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
 - B. The correct syntax uses `contains(@id, 'user')`.
- Incorrect Options Feedback:
 - o A. Matches only exact IDs.
 - o C. Matches prefix values.
 - o D. XPath does not have a direct `ends-with()` function.

Multiple Choice Question 3

How is `contains()` beneficial for locating dynamic text in automation?

Options:

- A. Reduces test code length
- B. Enables flexible matching when text updates slightly
- C. Matches only static headings
- D. Simplifies driver initialization

Skill: XPath

Subskill: Dynamic Text Recognition

Competency: Manage changing interface text

Difficulty Level: Intermediate

Bloom Level: Evaluation

- Correct Answer Reason:
 - B. It tolerates small text shifts in dynamic UI updates.
 - Incorrect Options Feedback:
 - o A. Code brevity is a side effect, not purpose.
 - o C. Dynamic handling is lost with fixed text.
 - o D. Driver setup is independent.
-

Multiple Choice Question 4

In terms of runtime efficiency, overusing `contains()` might cause:

Options:

- A. Improved locator stability
- B. Inconsistent performance on large DOMs
- C. Database lock
- D. Elimination of context switching

Skill: XPath

Subskill: Optimization

Competency: Balance flexibility and performance

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
 - B. `contains()` broadens searches, increasing lookup time.
 - Incorrect Options Feedback:
 - o A. Too many flexible locators harm precision.
 - o C. Irrelevant in UI automation.
 - o D. Context switching relates to frames or windows.
-

Multiple Choice Question 5

Which best practice improves test robustness using `contains()` locators?

Options:

- A. Combine with visible text checks
- B. Apply without attributes
- C. Use only with static IDs
- D. Avoid all conditional filters

Skill: XPath

Subskill: Robust Locator Design

Competency: Implement resilient automation

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Pairing `contains()` with visible text verification ensures accurate match context.
- Incorrect Options Feedback:
 - o B. Attributes provide stability.
 - o C. Static IDs make dynamic checks redundant.
 - o D. Conditions refine locator validity.

Advanced XPath: starts-with()

Multiple Choice Question 1

Which expression correctly identifies elements where the attribute “name” begins with “login”?

Options:

- A. `//*[@name='login']`
- B. `//*[starts-with(@name, 'login')]`
- C. `//*[contains(@name, 'login')]`
- D. `//*[text()='login']`

Skill: XPath

Subskill: Prefix-based Attribute Matching

Competency: Implement `starts-with()` effectively in locators

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. The `starts-with()` function finds elements whose attribute values begin with a specified prefix.
 - Incorrect Options Feedback:
 - o A. Matches exact attribute values only.
 - o C. Matches partial values, not necessarily at the start.
 - o D. Matches fixed text, not attributes.
-

Multiple Choice Question 2

What is the main advantage of using `starts-with()` over `contains()` in XPath expressions?

Options:

- A. Better matches for static identifiers
- B. Increased accuracy when attributes have consistent prefixes
- C. Reduced query flexibility
- D. Improved browser compatibility

Skill: XPath

Subskill: Function Selection

Competency: Select the proper XPath helper function

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
 - B. `starts-with()` ensures precision where attribute patterns consistently begin with a specific prefix.
- Incorrect Options Feedback:
 - o A. It's designed for partial, not exact, matches.

- o C. It maintains flexibility but with positional constraint.
 - o D. Browser compatibility is unrelated to XPath logic.
-

Multiple Choice Question 3

Which use of `starts-with()` ensures reliable element selection despite dynamically generated suffixes?

Options:

- A. `//*[@starts-with(@id, 'btn_')]`
- B. `//*[@contains(@id, '_btn')]`
- C. `//*[@id='btn_login']`
- D. `//*[@id='button']`

Skill: XPath

Subskill: Dynamic ID Recognition

Competency: Manage UI elements with predictable prefixes

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - A. The `@id` values that begin with “`btn_`” can be reliably located even if suffixes vary dynamically.
 - Incorrect Options Feedback:
 - o B. This would match `'_btn'` anywhere in the text.
 - o C. Targets only one exact ID.
 - o D. Generic and non-dynamic.
-

Multiple Choice Question 4

Overusing `starts-with()` in large web applications can negatively impact which aspect of test execution?

Options:

- A. Assertion logic
- B. DOM traversal performance
- C. Java code readability
- D. WebDriver version compatibility

Skill: XPath

Subskill: Optimization and Efficiency

Competency: Assess runtime performance of XPath strategies

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - B. Complex prefix-based searches increase DOM traversal time, reducing execution speed.

- Incorrect Options Feedback:
 - o A. Assertions happen after locator resolution.
 - o C. XPath affects test logic, not Java syntax.
 - o D. WebDriver APIs handle XPath uniformly across versions.
-

Multiple Choice Question 5

What is the most effective practice for using `starts-with()` in automation frameworks with changing UI prefixes?

Options:

- A. Combine `starts-with()` with stable ancestor context
- B. Use random waits alongside it
- C. Avoid it entirely in dynamic environments
- D. Replace it with tag-only locators

Skill: XPath

Subskill: Locator Robustness

Competency: Build maintainable and context-aware locators

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Combining prefix-based functions with context-specific parent or ancestor tags improves reliability and precision.
 - Incorrect Options Feedback:
 - o B. Timing mechanisms do not improve locator accuracy.
 - o C. It remains useful if scoped properly.
 - o D. Tag-only locators lose specificity.
-

Advanced XPath: axes (parent, following-sibling, ancestor)

Multiple Choice Question 1

Which XPath expression correctly selects the immediate parent element of a targeted child element?

Options:

- A. `//child::parent`
- B. `//element/parent::*`
- C. `//parent::element`
- D. `//ancestor::element`

Skill: XPath

Subskill: DOM Navigation using Axes

Competency: Navigate hierarchical relationships in DOM

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
B. The `parent::*` axis moves one level up to the parent of the current node.
 - Incorrect Options Feedback:
 - o A. The child axis moves downward, not upward.
 - o C. Incorrect use of `parent::` syntax.
 - o D. `ancestor::` includes all upper levels, not only the immediate parent.
-

Multiple Choice Question 2

When would the `following-sibling` axis be most useful in automation element targeting?

Options:

- A. When locating a hidden parent container
- B. When identifying elements that appear sequentially at the same hierarchy level
- C. When fetching descendant nodes deep in the tree
- D. When switching between frames

Skill: XPath

Subskill: DOM Traversal Optimization

Competency: Apply accurate sibling navigation

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
B. The `following-sibling` axis enables locating nodes at the same DOM level that appear after the current node.
 - Incorrect Options Feedback:
 - o A. Parent containers are accessed via `parent::`.
 - o C. Descendants use `descendant::`, not sibling axes.
 - o D. Frame handling is unrelated to XPath axes.
-

Multiple Choice Question 3

Which XPath correctly identifies an element's ancestor with a specific tag name?

Options:

- A. `//ancestor::div`
- B. `//div/ancestor::*`
- C. `//*[@id='item']/ancestor::section`
- D. `//child::ancestor`

Skill: XPath

Subskill: Ancestor and Hierarchical Search

Competency: Retrieve higher-level nodes conditionally

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
C. The `ancestor::section` axis retrieves ancestor nodes of the specified element that match a given tag.
 - Incorrect Options Feedback:
 - o A. Invalid context because it targets all div ancestors globally.
 - o B. Reversed order of current node and axis.
 - o D. `child::ancestor` is syntactically invalid.
-

Multiple Choice Question 4

In complex document structures, what is a key performance consideration when using multiple XPath axes such as `ancestor` and `following-sibling`?

Options:

- A. They always execute faster than CSS selectors
- B. They can lead to broader DOM traversals and slow lookups
- C. They are ignored by modern browsers
- D. They cannot be combined in one XPath

Skill: XPath

Subskill: DOM Efficiency

Competency: Optimize XPath for large trees

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
B. Using multiple axes broadens the search scope, which can increase evaluation time on large documents.
 - Incorrect Options Feedback:
 - o A. Axes are typically slower than optimized CSS selectors.
 - o C. Browsers process them normally.
 - o D. Multiple axes can be combined logically.
-

Multiple Choice Question 5

What is the most maintainable approach for using axes-based XPaths in long-term automation frameworks?

Options:

- A. Limit usage by combining them with attribute or text filters
- B. Use axes for every locator uniformly
- C. Replace all axes with full CSS selectors
- D. Define axes without conditions for faster execution

Skill: XPath

Subskill: Maintainability and Precision

Competency: Apply sustainable XPath strategies

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Combining axes with attributes or conditions avoids fragile deep DOM dependencies while preserving flexibility.
 - Incorrect Options Feedback:
 - o B. Overuse increases complexity.
 - o C. CSS selectors lack hierarchical reach in some cases.
 - o D. Unfiltered axes are inefficient and error-prone.
-

Creating XPath and CSS Locators for Dynamic Elements

Multiple Choice Question 1

When dealing with elements that change IDs between page loads, which locator approach is most reliable?

Options:

- A. Use fixed ID values from the previous session
- B. Use XPath with partial attribute matches
- C. Depend solely on tag names
- D. Disable ID validation in the framework

Skill: Dynamic Locators

Subskill: XPath Partial Matching

Competency: Identify effective strategies for unstable attributes

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. XPath partial matches handle variable parts of IDs while keeping locator accuracy.
 - Incorrect Options Feedback:
 - o A. Fixed values break as IDs change dynamically.
 - o C. Tags alone are generic and prone to collisions.
 - o D. Disabling validation risks locator mismatches.
-

Multiple Choice Question 2

Which CSS selector effectively targets an element whose class includes a dynamic substring “_active”?

Options:

- A. .active

- B. `[class='_active']`
- C. `[class*='_active']`
- D. `[class^='_active']`

Skill: CSS Selectors

Subskill: Attribute Wildcards

Competency: Design flexible CSS locators for varying values

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
C. The asterisk (*) operator matches strings containing the specified substring, ideal for dynamic parts.
 - Incorrect Options Feedback:
 - o A. Matches only static class names.
 - o B. Requires an exact match.
 - o D. Matches only prefixes, not substrings.
-

Multiple Choice Question 3

Which locator strategy reduces flakiness in tests involving dynamic lists with IDs that include timestamps?

Options:

- A. Use `starts-with()` in XPath with static prefix
- B. Use absolute XPath
- C. Match the entire ID dynamically with regex
- D. Avoid using any attributes

Skill: XPath

Subskill: Stable Locator Creation

Competency: Manage dynamically generated attributes

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
A. The static portion of dynamic IDs can be referenced with `starts-with()`, allowing robust element targeting.
 - Incorrect Options Feedback:
 - o B. Absolute XPath are brittle.
 - o C. Selenium does not support inline regex matching directly.
 - o D. Attribute-based locators are essential for precision.
-

Multiple Choice Question 4

When crafting a CSS locator for dynamic elements, what best practice improves maintainability?

Options:

- A. Combine multiple class and attribute conditions
- B. Use inline text selectors only
- C. Use generated DOM indexes
- D. Apply universal (*) selectors widely

Skill: CSS Selectors

Subskill: Maintainable Locators

Competency: Construct efficient and readable CSS locators

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - A. Combining multiple stable class and attribute rules enhances targeting accuracy and resilience.
 - Incorrect Options Feedback:
 - o B. Inline text is not supported in CSS selection.
 - o C. Index-based locators are fragile.
 - o D. Universal selectors increase performance overhead.
-

Multiple Choice Question 5

What technique is most effective for detecting dynamic content that loads after user interaction (e.g., AJAX updates) before locating it?

Options:

- A. Combine dynamic locators with explicit waits for visibility
- B. Use static locators with Thread.sleep()
- C. Locate elements before initiating page interaction
- D. Disable implicit waiting for optimization

Skill: Dynamic Element Handling

Subskill: Synchronization + Locator Integration

Competency: Manage dynamic element visibility and readiness

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Explicit waits ensure the locator resolves only once the dynamic content is ready for interaction.
 - Incorrect Options Feedback:
 - o B. Fixed delay is unreliable.
 - o C. Premature location leads to stale element errors.
 - o D. Disabling waits compromises consistency.
-

Synchronization Challenges in Automation (Why Waits Are Needed)

Multiple Choice Question 1

What is the most common root cause of synchronization issues in Selenium test automation?

Options:

- A. Incorrect data providers
- B. Variability between element load time and script execution speed
- C. Wrong browser capabilities
- D. Mismatched version of ChromeDriver

Skill: Synchronization

Subskill: Timing Discrepancies

Competency: Identify root causes of flaky scripts

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - B. Synchronization problems occur when test scripts attempt interactions before elements fully load.
 - Incorrect Options Feedback:
 - o A. Data providers relate to input handling, not synchronization.
 - o C. Desired capabilities affect session configuration, not load timing.
 - o D. Version mismatch affects compatibility, not timing.
-

Multiple Choice Question 2

Why are waits critical for handling AJAX-heavy applications in Selenium tests?

Options:

- A. Waits help achieve parallel testing
- B. Waits monitor JavaScript-driven asynchronous updates
- C. Waits improve browser speed
- D. Waits reduce object locators count

Skill: Synchronization

Subskill: AJAX Handling

Competency: Manage asynchronous content loading

Difficulty Level: Advanced

Bloom Level: Comprehension

- Correct Answer Reason:
 - B. Waits handle dynamically loading elements in AJAX-based UIs by delaying interactions until content becomes stable.
- Incorrect Options Feedback:
 - o A. Parallel testing involves threading, not waits.

- o C. Waits add synchronization time rather than speed.
 - o D. The locator count remains the same.
-

Multiple Choice Question 3

If tests intermittently fail due to elements not being clickable or visible, what synchronization improvement should be prioritized?

Options:

- A. Introduce explicit waits before interactions
- B. Add more assertions
- C. Use Thread.sleep() throughout the test
- D. Rerun the failed test cases without changes

Skill: Synchronization Strategy

Subskill: Wait Optimization

Competency: Stabilize interactive element readiness

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Explicit waits address element readiness precisely by pausing execution until clickability or visibility conditions are met.
 - Incorrect Options Feedback:
 - o B. Assertions validate, not synchronize.
 - o C. Sleep statements are inefficient and unreliable.
 - o D. Rerunning without correction repeats failure.
-

Multiple Choice Question 4

Which synchronization concept helps align automation speed with user experience delays like transitions or animations?

Options:

- A. Script timeout
- B. Explicit waits with ExpectedConditions
- C. Headless browser execution
- D. Static wait implementation

Skill: Automation Synchronization

Subskill: UX Delay Handling

Competency: Match script execution to UI transition timing

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. Explicit waits using ExpectedConditions effectively time interactions with visual UI transitions and loading.

- Incorrect Options Feedback:
 - o A. Script timeouts handle async scripts, not UI delays.
 - o C. Headless mode reduces animations but doesn't synchronize.
 - o D. Static waits waste time and reduce scalability.
-

Multiple Choice Question 5

What is the recommended method to ensure synchronization reliability in cross-browser automation environments?

Options:

- A. Combine implicit and explicit waits dynamically
- B. Use explicit waits with robust ExpectedConditions definitions
- C. Replace waits entirely with retry loops
- D. Disable synchronization across browsers

Skill: Cross-Browser Automation

Subskill: Reliable Wait Management

Competency: Ensure portable synchronization strategies

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - B. Explicit waits are browser-agnostic and provide fine control over dynamic behavior across different engines.
 - Incorrect Options Feedback:
 - o A. Combining waits may cause unexpected timing conflicts.
 - o C. Retry loops lack Selenium's built-in timeout handling.
 - o D. Disabling synchronization leads to frequent flakiness.
-

Implicit Waits

Multiple Choice Question 1

What is the key characteristic of an implicit wait in Selenium WebDriver?

Options:

- A. It waits only for specific elements when defined
- B. It applies globally to all element lookups
- C. It handles both AJAX and static content separately
- D. It supports waiting for custom conditions

Skill: Synchronization Mechanisms

Subskill: Implicit Wait Fundamentals

Competency: Identify global waiting behavior

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
B. Implicit waits apply globally, affecting all subsequent element-finding operations in a WebDriver session.
 - Incorrect Options Feedback:
 - o A. It is not element-specific.
 - o C. AJAX handling requires explicit waits.
 - o D. Custom conditions are implemented using explicit waits.
-

Multiple Choice Question 2

In Selenium, how does implicit wait affect the `findElement()` method?

Options:

- A. It retries locating the element until the wait time expires or the element appears
- B. It executes once regardless of element state
- C. It immediately throws a `NoSuchElementException`
- D. It forces the browser to reload elements repeatedly

Skill: Implicit Wait

Subskill: Behavior on Element Search

Competency: Manage element discovery retries

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
A. Implicit wait causes Selenium to retry locating an element within the defined timeout before throwing an exception.
 - Incorrect Options Feedback:
 - o B. Implicit wait is continuously active within the timeout window.
 - o C. The exception appears only after the period expires.
 - o D. Reloading the page is not part of implicit waiting.
-

Multiple Choice Question 3

Which situation illustrates an inappropriate use of an implicit wait?

Options:

- A. Short waits to handle infrequent network delays
- B. Complex applications with frequent asynchronous updates
- C. Static pages with predictable load times
- D. Pages where all elements render simultaneously

Skill: Synchronization Strategy

Subskill: Implicit Wait Limitations

Competency: Identify scenarios for better synchronization choice

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
B. Implicit waits cannot handle frequent asynchronous or conditional visibility changes common in complex dynamic applications.
 - Incorrect Options Feedback:
 - o A. Mild variability is acceptable.
 - o C. Predictable pages suit implicit waits.
 - o D. Fully static pages need minimal synchronization.
-

Multiple Choice Question 4

What are the side effects of combining implicit and explicit waits in a Selenium script?

Options:

- A. Script executes faster due to stacked waits
- B. Wait time multiplies unpredictably, leading to longer delays
- C. Each wait overrides the other cleanly
- D. WebDriver automatically resolves conflicts

Skill: Synchronization

Subskill: Wait Interactions

Competency: Handle overlapping wait configurations

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
B. Using both waits together can compound timeouts, causing performance issues.
 - Incorrect Options Feedback:
 - o A. Stacked waits delay execution, not speed it up.
 - o C. They do not override automatically.
 - o D. No built-in mechanism resolves these conflicts.
-

Multiple Choice Question 5

What is the correct syntax for setting an implicit wait in Selenium using Java?

Options:

- A. `driver.wait(10);`
- B. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));`
- C. `driver.manage().window().wait(10);`
- D. `driver.setWait(10);`

Skill: Selenium Configuration

Subskill: Implicit Wait Implementation

Competency: Implement synchronization at session level

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
B. The correct Java syntax uses the `manage().timeouts()` method with `implicitlyWait()` to set a global timeout.
 - Incorrect Options Feedback:
 - o A. The `WebDriver` interface lacks a direct `wait()` call.
 - o C. The `window()` method controls browser size, not timeouts.
 - o D. `setWait()` is not a valid Selenium function.
-

Explicit Waits (WebDriverWait, ExpectedConditions)

Multiple Choice Question 1

What key advantage does an explicit wait provide over an implicit wait in Selenium?

Options:

- A. It applies automatically to every element lookup
- B. It waits for specific conditions before proceeding
- C. It uses a fixed delay regardless of element state
- D. It replaces thread synchronization features

Skill: Synchronization

Subskill: Conditional Waiting

Competency: Apply explicit synchronization effectively

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
B. Explicit waits pause execution based on defined conditions like visibility, clickability, or presence of elements.
 - Incorrect Options Feedback:
 - o A. Implicit waits, not explicit, are global.
 - o C. Explicit waits terminate early when conditions are met.
 - o D. They do not replace thread-level synchronization.
-

Multiple Choice Question 2

Which class in Selenium provides the core structure for defining explicit waits?

Options:

- A. `WaitHandler`
- B. `FluentWait`
- C. `WebDriverWait`
- D. `WebElementWait`

Skill: Explicit Waits

Subskill: Class Implementation

Competency: Utilize proper Java classes for synchronization

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
C. `WebDriverWait` is a Selenium wrapper that simplifies explicit wait usage around conditional waiting logic.
 - Incorrect Options Feedback:
 - o A. `WaitHandler` does not exist.
 - o B. `FluentWait` is a parent class with more customization.
 - o D. `WebElementWait` is invalid in `WebDriver`'s API.
-

Multiple Choice Question 3

What happens when the `ExpectedCondition` in an explicit wait is not satisfied within the timeout duration?

Options:

- A. The command executes with null value
- B. `WebDriver` throws a `TimeoutException`
- C. The condition resets automatically
- D. The driver quits the session

Skill: Synchronization

Subskill: Timeout Management

Competency: Handle exception flow in explicit waits

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
B. If a condition fails within the timeout, Selenium throws `TimeoutException` indicating synchronization failure.
 - Incorrect Options Feedback:
 - o A. Explicit waits do not return nulls on failure.
 - o C. Conditions do not reset automatically.
 - o D. Driver sessions remain active after such exceptions.
-

Multiple Choice Question 4

Which `ExpectedCondition` should be used to confirm that a web element is visible on the page before interaction?

Options:

- A. `elementToBeClickable()`
- B. `presenceOfElementLocated()`

- C. `visibilityOfElementLocated()`
- D. `invisibilityOfElementLocated()`

Skill: Automation Waits

Subskill: ExpectedConditions Functions

Competency: Select suitable visibility verification condition

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
`C.visibilityOfElementLocated()` verifies that an element is both present and visible for interaction.
 - Incorrect Options Feedback:
 - o A. Checks for clickability, not visibility.
 - o B. Only ensures presence in the DOM.
 - o D. Used for waiting until an element disappears.
-

Multiple Choice Question 5

What is a key design practice when integrating explicit waits into a Page Object Model (POM)?

Options:

- A. Defining explicit waits directly inside test scripts
- B. Centralizing wait logic within page methods
- C. Using waits globally across all pages
- D. Replacing waits with constant sleeps

Skill: Selenium Design

Subskill: Page Object Synchronization

Competency: Integrate waits cleanly in maintainable frameworks

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
B. Wait logic should reside in page methods for encapsulation and better maintainability of the POM structure.
 - Incorrect Options Feedback:
 - o A. Mixing waits in test scripts breaks abstraction.
 - o C. Explicit waits shouldn't apply globally.
 - o D. Static sleeps reduce efficiency and precision.
-

Fluent Waits

Multiple Choice Question 1

What distinguishes a Fluent Wait from an Explicit Wait in Selenium?

Options:

- A. Explicit waits allow defining polling frequency and ignoring conditions
- B. Fluent waits allow customizing polling intervals and exception handling
- C. Fluent waits are globally configured like implicit waits
- D. Explicit waits execute faster than Fluent waits

Skill: Synchronization

Subskill: Wait Configuration

Competency: Understand difference between explicit and fluent waits

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - B. Fluent waits extend explicit waits with control over polling rates and ignored exceptions, offering flexible synchronization.
 - Incorrect Options Feedback:
 - o A. Polling adjustment is unique to Fluent waits.
 - o C. Fluent waits are instance-specific, not global.
 - o D. Execution speed depends on timeout and polling logic, not type.
-

Multiple Choice Question 2

Which Fluent Wait method specifies how often Selenium checks for the defined condition?

Options:

- A. `withTimeout()`
- B. `pollingEvery()`
- C. `ignoring()`
- D. `until()`

Skill: Wait Mechanics

Subskill: Polling Frequency Control

Competency: Define check intervals efficiently

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
 - B. The `pollingEvery()` method defines time intervals at which Selenium polls the DOM for expected conditions.
 - Incorrect Options Feedback:
 - o A. Sets the total duration.
 - o C. Handles exceptions, not polling.
 - o D. Executes the waiting condition.
-

Multiple Choice Question 3

Which scenario best demonstrates when to use Fluent Wait?

Options:

- A. When element appearance timing varies and exceptions need to be ignored during checking
- B. When all elements are static
- C. When using static delays
- D. When handling only single browser tabs

Skill: Synchronization

Subskill: Adaptive Waiting

Competency: Choose optimal wait strategy

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Fluent Wait is ideal for dynamic elements with uncertain load times and intermittent exceptions like `NoSuchElementException`.
 - Incorrect Options Feedback:
 - o B. Static elements require minimal waiting.
 - o C. Static waits lack flexibility.
 - o D. Browser tab count is unrelated to synchronization.
-

Multiple Choice Question 4

In Fluent Wait, which type of exception is commonly ignored to prevent test interruption during polling?

Options:

- A. `NullPointerException`
- B. `NoSuchElementException`
- C. `IOException`
- D. `StaleElementReferenceException`

Skill: Synchronization

Subskill: Exception Management

Competency: Improve test resilience during frequent retries

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. `NoSuchElementException` is often ignored in Fluent Wait to allow repeated attempts until elements appear.
- Incorrect Options Feedback:
 - o A. Null pointers are coding issues, not locator-related.
 - o C. `IOException` belongs to file operations.
 - o D. Stale element errors occur post-locating.

Multiple Choice Question 5

Which Fluent Wait configuration ensures efficient waiting without unnecessary resource usage?

Options:

- A. High timeout and low polling frequency combination
- B. Low timeout and random poll intervals
- C. Reasonable timeout matched with optimized polling interval and clear ignored exceptions
- D. Infinite wait with frequent polling

Skill: Synchronization Design

Subskill: Performance Optimization

Competency: Configure efficient waits

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - C. Balanced timeout and polling frequency ensure responsive checks while minimizing CPU use and wasted waiting.
 - Incorrect Options Feedback:
 - o A. Excessive timeout delays execution.
 - o B. Randomized polling is unpredictable.
 - o D. Infinite waits stall automation indefinitely.
-

Handling Synchronization in Real-World Scenarios

Multiple Choice Question 1

When scripts frequently fail because elements load asynchronously across environments, what should be the first synchronization step?

Options:

- A. Add a fixed Thread.sleep() after every action
- B. Analyze and apply custom explicit waits for critical elements
- C. Disable synchronization during retesting
- D. Increase implicit wait globally to maximum duration

Skill: Synchronization Strategy

Subskill: Real-world Debugging

Competency: Implement targeted wait configurations

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
B. Applying explicit waits selectively for dynamic elements aligns execution with specific load conditions efficiently.
 - Incorrect Options Feedback:
 - o A. Fixed sleeps are inefficient and can mask issues.
 - o C. Disabling synchronization worsens instability.
 - o D. Long implicit waits delay all test steps unnecessarily.
-

Multiple Choice Question 2

In an application where animation delays vary, how should synchronization be optimized?

Options:

- A. Turn off animations using JavaScriptExecutor and wait for stable states
- B. Use only implicit waits
- C. Execute actions immediately after element presence
- D. Increase the polling interval to reduce animation detection

Skill: Synchronization

Subskill: UI Animation Handling

Competency: Manage dynamic interface transitions

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
A. Disabling animations or using JavaScript-assisted waits ensures that elements are stable before interaction.
 - Incorrect Options Feedback:
 - o B. Implicit waits do not manage dynamic motion.
 - o C. Actions may fail during transitions.
 - o D. Long polling skips important timing checks.
-

Multiple Choice Question 3

How can synchronization enhance data-driven testing in rapidly updating web forms?

Options:

- A. By re-entering data multiple times automatically
- B. By stabilizing field availability before performing sendKeys actions
- C. By increasing element lookup speed
- D. By caching values from previous runs

Skill: Synchronization in Data Automation

Subskill: Input Element Reliability

Competency: Ensure interaction readiness for data inputs

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
B. Synchronization ensures that the targeted fields are interactable before performing data entry actions.
 - Incorrect Options Feedback:
 - o A. Re-entry wastes time.
 - o C. Lookup speed depends on DOM, not waits.
 - o D. Caching does not resolve readiness issues.
-

Multiple Choice Question 4

Which combination of techniques is best suited for handling synchronization in large-scale parallel Selenium test suites?

Options:

- A. Global implicit waits and random sleeps
- B. Explicit waits combined with well-scoped Page Object methods
- C. Custom retry loops added after each wait
- D. Browser refresh after every failed element lookup

Skill: Synchronization Design

Subskill: Framework Scalability

Competency: Design scalable and maintainable synchronization models

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
B. Integrating explicit waits inside POM methods keeps synchronization localized and reusable across multiple parallel tests.
 - Incorrect Options Feedback:
 - o A. Random sleeps cause performance inefficiency.
 - o C. Retries without logic cause redundant processing.
 - o D. Browser refreshes mask timing errors.
-

Multiple Choice Question 5

When synchronizing API-driven web components that render asynchronously, what approach should a tester adopt?

Options:

- A. Wait for API response completion via JavaScriptExecutor checking readiness states
- B. Use long implicit waits
- C. Execute next step without synchronization
- D. Rely only on element count validation

Skill: Test Automation

Subskill: Hybrid Synchronization

Competency: Synchronize front-end automation with backend API states

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
A. JavaScriptExecutor can check backend responses or DOM readiness flags before UI interaction, aligning with asynchronous behavior.
 - Incorrect Options Feedback:
 - o B. Implicit waits are ineffective for asynchronous API rendering.
 - o C. Ignoring synchronization causes element-not-found errors.
 - o D. Element count validation alone doesn't ensure readiness.
-

Selenium Setup and Basics

Multiple Choice Question 1

Which step is essential before executing a Selenium WebDriver script on a specific browser?

Options:

- A. Installing Java Runtime Environment only
- B. Configuring the corresponding browser driver executable
- C. Downloading all Selenium framework modules
- D. Enabling network proxy for the browser

Skill: Selenium Setup

Subskill: Environment Configuration

Competency: Prepare execution prerequisites for Selenium tests

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
B. Selenium requires a WebDriver binary (e.g., ChromeDriver, GeckoDriver) to interface with browsers.
 - Incorrect Options Feedback:
 - o A. JRE alone doesn't enable browser automation.
 - o C. Not all modules are mandatory for basic setup.
 - o D. Proxy configuration is optional for special environments.
-

Multiple Choice Question 2

Which Selenium component acts as the interface between test scripts and browsers?

Options:

- A. Selenium Grid
- B. Selenium IDE
- C. Selenium WebDriver
- D. Selenium RC

Skill: Selenium Basics

Subskill: Architecture Understanding

Competency: Identify roles of Selenium components

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
C. Selenium WebDriver communicates directly with browser drivers to execute commands.
 - Incorrect Options Feedback:
 - o A. Selenium Grid distributes tests, not execution commands.
 - o B. Selenium IDE is a recording tool.
 - o D. Selenium RC is deprecated.
-

Multiple Choice Question 3

In Selenium architecture, what is the role of the browser-specific driver executable (e.g., ChromeDriver)?

Options:

- A. Parses web pages for tests automatically
- B. Translates JSON wire protocol commands to native browser instructions
- C. Performs DOM assertions
- D. Logs browser cookies for test reporting

Skill: Selenium Architecture

Subskill: Command Translation

Competency: Understand communication flow between WebDriver and browsers

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
B. The driver acts as a translator that converts WebDriver API calls into browser-native commands.
 - Incorrect Options Feedback:
 - o A. Parsing is handled by test scripts.
 - o C. Assertions are part of test logic, not WebDriver.
 - o D. Cookie logging is optional and separate.
-

Multiple Choice Question 4

During Selenium setup, which step ensures compatibility between the WebDriver and the browser?

Options:

- A. Matching driver version to Java JDK version
- B. Using the WebDriverManager utility or manually updating drivers according to browser versions

- C. Setting JAVA_HOME path variable
- D. Disabling browser extensions

Skill: Environment Setup

Subskill: Version Management

Competency: Maintain synchronized browser-driver versions

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. Selenium tests require driver versions compatible with the browser; tools like WebDriverManager automate this process.
 - Incorrect Options Feedback:
 - o A. The driver's compatibility is independent of Java.
 - o C. Path variables are needed for Java, not browser compatibility.
 - o D. Extensions do not affect driver binding.
-

Multiple Choice Question 5

After setting up Selenium WebDriver, what command validates the environment configuration?

Options:

- A. `webdriver.startSession()`
- B. `driver.get("https://example.com")`
- C. `driver.findElement(By.id("test"))`
- D. `driver.launchBrowser()`

Skill: Selenium Configuration Validation

Subskill: Test Initialization

Competency: Confirm Selenium setup success

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. Executing `driver.get()` opens a target URL, validating correct driver initialization and browser connection.
 - Incorrect Options Feedback:
 - o A. No such method exists in Selenium API.
 - o C. Requires a prior session to run correctly.
 - o D. `launchBrowser()` is not part of WebDriver syntax.
-

Selenium Components and Architecture

Multiple Choice Question 1

Which Selenium component provides the capability to execute tests across multiple machines and browsers simultaneously?

Options:

- A. Selenium IDE
- B. Selenium Grid
- C. Selenium WebDriver
- D. Selenium RC

Skill: Selenium Architecture

Subskill: Parallel Test Execution

Competency: Identify distributed testing tools within Selenium

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
 - B. Selenium Grid manages distributed execution by running tests across different nodes and environments concurrently.
 - Incorrect Options Feedback:
 - o A. Selenium IDE supports record and playback only.
 - o C. WebDriver runs tests locally.
 - o D. Selenium RC is deprecated and not distributed by default.
-

Multiple Choice Question 2

How does Selenium WebDriver interact with browsers internally?

Options:

- A. By executing shell commands
- B. Through the JSON Wire Protocol or W3C WebDriver protocol
- C. Using event listeners inside test classes
- D. By embedding scripts directly into HTML pages

Skill: Selenium Architecture

Subskill: Communication Protocols

Competency: Understand internal command execution mechanisms

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
 - B. Selenium WebDriver communicates browser instructions using the JSON Wire or W3C protocol standards.
- Incorrect Options Feedback:
 - o A. WebDriver uses API calls, not OS shell commands.

- o C. Event listeners belong to Java test frameworks.
 - o D. It does not alter page HTML directly.
-

Multiple Choice Question 3

What primary purpose does the Selenium Server serve in a Grid architecture?

Options:

- A. Acts as a proxy between WebDriver and browsers
- B. Stores test scripts in memory
- C. Handles test reporting
- D. Compiles WebDriver class files

Skill: Selenium Architecture

Subskill: Hub-Node Communication

Competency: Understand the role of Selenium Server

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - A. The Selenium Server Hub coordinates communication between WebDriver clients and registered browser nodes.
 - Incorrect Options Feedback:
 - o B. Test scripts are stored externally.
 - o C. Reporting is handled by frameworks like TestNG.
 - o D. WebDriver already executes compiled Java code.
-

Multiple Choice Question 4

In Selenium's layered architecture, which component serves as the core interface for element actions like click or sendKeys?

Options:

- A. WebDriver API
- B. Browser Engine
- C. Selenium Grid
- D. JSON Wire Protocol

Skill: Selenium Basics

Subskill: API Interaction

Competency: Identify functional interaction layer

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - A. The WebDriver API provides high-level methods like click() and sendKeys() for interacting with page elements.

- Incorrect Options Feedback:
 - o B. The browser engine executes received commands; it's not the API layer.
 - o C. Selenium Grid focuses on parallel execution only.
 - o D. JSON Wire Protocol transfers commands but is not where interactions occur.
-

Multiple Choice Question 5

What is the correct sequence of interaction in the Selenium automation flow?

Options:

- A. Test Script → Browser → WebDriver → DOM
- B. Browser → WebDriver → Selenium Server → Test Script
- C. Test Script → WebDriver → Browser Driver → Browser
- D. WebDriver → Test Script → Selenium Grid → Browser

Skill: Selenium Architecture

Subskill: Command Flow Understanding

Competency: Explain Selenium execution lifecycle

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
 - C. Selenium interacts in this order: Test Script issues commands → WebDriver → Browser Driver → Browser executes actions.
 - Incorrect Options Feedback:
 - o A. Command order is reversed.
 - o B. Selenium Server is optional.
 - o D. Grid is relevant only for remote execution.
-

Launching Browsers (Chrome, Firefox, Edge) with Selenium WebDriver

Multiple Choice Question 1

Which of the following is required to execute Selenium tests on Chrome browser successfully?

Options:

- A. Setting System property for ChromeDriver with browser binary path
- B. Installing Firefox add-ons
- C. Enabling autosave settings in Chrome
- D. Enabling DOM Storage manually

Skill: Browser Launching

Subskill: ChromeDriver Setup

Competency: Configure browser-specific WebDriver environment

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
A. Selenium requires setting the ChromeDriver system property or using WebDriverManager to connect Chrome with the WebDriver API.
 - Incorrect Options Feedback:
 - o B. Firefox add-ons are irrelevant for Chrome.
 - o C. Autosave has no effect on WebDriver connection.
 - o D. DOM Storage is browser-managed by default.
-

Multiple Choice Question 2

Which constructor initializes the Firefox browser using Selenium WebDriver in Java?

Options:

- A. `WebDriver driver = new GeckoDriver();`
- B. `WebDriver driver = new FirefoxDriver();`
- C. `WebDriver driver = new FirefoxWebDriver();`
- D. `WebDriver driver = new GeckoFirefoxDriver();`

Skill: Selenium Browser Setup

Subskill: Firefox Launch

Competency: Initialize correct driver objects for browser tests

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
B. Instantiating `FirefoxDriver()` initiates the `GeckoDriver` process, which launches Firefox browser sessions.
 - Incorrect Options Feedback:
 - o A, C, D are invalid constructors in Selenium's API.
-

Multiple Choice Question 3

When automating on Microsoft Edge using Selenium, which driver needs to be configured?

Options:

- A. `IEDriverServer`
- B. `EdgeDriver`
- C. `WebKitDriver`
- D. `ChromiumGridDriver`

Skill: Browser Automation

Subskill: Microsoft Edge Configuration

Competency: Recognize appropriate driver for different browsers

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
B. EdgeDriver is required for automation with Microsoft Edge, compatible with the Chromium architecture.
 - Incorrect Options Feedback:
 - o A. IEDriver applies to older Internet Explorer versions.
 - o C. WebKitDriver is not part of Selenium.
 - o D. ChromiumGridDriver doesn't exist.
-

Multiple Choice Question 4

Which method is called to launch a specific URL after initializing the WebDriver?

Options:

- A. `driver.load("url");`
- B. `driver.start("url");`
- C. `driver.get("url");`
- D. `driver.openURL("url");`

Skill: WebDriver Basics

Subskill: Browser Navigation

Competency: Use WebDriver command to open web pages

Difficulty Level: Beginner to Intermediate

Bloom Level: Application

- Correct Answer Reason:
C. The `get()` method loads a web page into the current browser session initiated by WebDriver.
 - Incorrect Options Feedback:
 - o A, B, D are invalid method names in Selenium's WebDriver API.
-

Multiple Choice Question 5

When launching browsers on remote machines using Selenium Grid, which protocol handles the communication between client and nodes?

Options:

- A. HTTP over JSON Wire Protocol
- B. FTP protocol
- C. TCP with SOAP requests
- D. WebSocket encryption layer

Skill: Distributed Selenium

Subskill: Grid Communication

Competency: Manage browser session creation remotely

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
A. Selenium Grid uses HTTP-based JSON Wire Protocol to transmit commands between the Hub and browser nodes.
 - Incorrect Options Feedback:
 - o B. FTP is used for file transfer, not command dispatch.
 - o C. SOAP is unrelated.
 - o D. WebSocket is not part of Selenium Grid communication.
-

Locating Elements: ID, Name, Class, TagName, LinkText

Multiple Choice Question 1

Which locator strategy provides the fastest and most reliable element identification method in Selenium?

Options:

- A. Class name
- B. ID
- C. Tag name
- D. Link text

Skill: Element Identification

Subskill: Locator Efficiency

Competency: Choose optimal locator strategies

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
B. Locating by ID is the fastest since it directly maps to a unique DOM property without extra parsing.
 - Incorrect Options Feedback:
 - o A. Class names may not always be unique.
 - o C. Tag names are too generic.
 - o D. Link text is restricted to anchor tags.
-

Multiple Choice Question 2

Which Selenium method locates an element using its HTML “name” attribute?

Options:

- A. `findElement(By.name("username"))`
- B. `findElement(By.id("username"))`
- C. `findElement(By.tag("username"))`
- D. `findElement(By.text("username"))`

Skill: Web Element Interaction
Subskill: Locator Methods
Competency: Use correct WebDriver locator syntax
Difficulty Level: Beginner to Intermediate
Bloom Level: Application

- Correct Answer Reason:
A. The `By.name()` method identifies elements using the name attribute value.
 - Incorrect Options Feedback:
 - o B. Searches using ID instead of name.
 - o C. `By.tag` is invalid in Selenium API.
 - o D. Selenium does not use `By.text()` as a locator.
-

Multiple Choice Question 3

Which locator strategy is best suited when multiple elements share the same class name but are differentiated by their tag structure or relative position?

Options:

- A. Class name combined with tag name or XPath
- B. Implicit wait usage
- C. Partial link text reference
- D. Global CSS selector

Skill: Locator Optimization
Subskill: Multi-attribute Targeting
Competency: Develop precise locator definitions
Difficulty Level: Advanced
Bloom Level: Evaluation

- Correct Answer Reason:
A. Using tag name or XPath in combination with class names refines element targeting among similar classes.
 - Incorrect Options Feedback:
 - o B. Waits handle timing, not identification.
 - o C. Link text applies only to anchors.
 - o D. Global selectors are less efficient and risk overmatching.
-

Multiple Choice Question 4

For hyperlink elements, when is `By.partialLinkText()` preferable over `By.linkText()`?

Options:

- A. When links have dynamic prefixes or suffixes
- B. When link text is static
- C. When HTML tags contain no `href` attribute
- D. When links require case-sensitive matching

Skill: Element Location
Subskill: Link Targeting
Competency: Apply link-based locator strategies
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason:
A. Partial link text locators handle dynamic or truncated link texts, improving flexibility.
 - Incorrect Options Feedback:
 - o B. Full static texts use `By.linkText()`.
 - o C. Both link methods require `href`.
 - o D. Link locators are not case-sensitive.
-

Multiple Choice Question 5

What is a primary disadvantage of using `By.tagName()` for element location?

Options:

- A. Slower execution time
- B. Limited browser compatibility
- C. Low specificity and potential for multiple matches
- D. Incompatibility with Selenium Grid

Skill: Locator Selection
Subskill: Tag-Level Identification
Competency: Recognize locator precision trade-offs
Difficulty Level: Intermediate
Bloom Level: Analysis

- Correct Answer Reason:
C. Tag names are often shared by multiple elements, making them less precise unless combined with other attributes.
 - Incorrect Options Feedback:
 - o A. Performance depends on DOM size, not locator type.
 - o B. It is fully compatible across browsers.
 - o D. Selenium Grid does not affect locators.
-

Element Interactions: `sendKeys`, `click`, `getText`, `clear`

Multiple Choice Question 1

What is the primary function of the `sendKeys()` method in Selenium WebDriver?

Options:

- A. Simulates typing into an input or editable field
- B. Performs a click on the target element

- C. Extracts visible text from an element
- D. Clears the value of a field

Skill: WebDriver Commands

Subskill: Input Handling

Competency: Use correct methods for element interaction

Difficulty Level: Beginner to Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
 - A. The `sendKeys()` method simulates user keyboard input by entering text into form fields or inputs.
 - Incorrect Options Feedback:
 - o B. `click()` is used for button or link activation.
 - o C. `getText()` retrieves existing content, not input.
 - o D. `clear()` removes current input from fields.
-

Multiple Choice Question 2

Which Selenium method would you use to retrieve text from a label or paragraph element displayed on a webpage?

Options:

- A. `getAttribute("value")`
- B. `getText()`
- C. `sendKeys("text")`
- D. `getVisibleText()`

Skill: Web Element Interaction

Subskill: Content Reading

Competency: Fetch displayed text dynamically

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. The `getText()` method captures visible text rendered on the page for the specified element.
 - Incorrect Options Feedback:
 - o A. `getAttribute()` extracts HTML attributes only.
 - o C. `sendKeys()` is for input.
 - o D. `getVisibleText()` is not an existing method.
-

Multiple Choice Question 3

What occurs if `click()` is called on an element that is hidden or not interactable yet?

Options:

- A. Selenium retries until visibility increases automatically

- B. A `ElementNotInteractableException` is thrown
- C. Selenium ignores the element and continues
- D. The element becomes visible automatically

Skill: Element Synchronization

Subskill: Action Preconditions

Competency: Handle visibility issues before user actions

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason:
B. Selenium throws `ElementNotInteractableException` if the targeted element cannot be clicked due to invisibility or overlap.
 - Incorrect Options Feedback:
 - o A. Retrying requires explicit waits.
 - o C. Execution stops when an exception occurs.
 - o D. Selenium does not auto-reveal hidden elements.
-

Multiple Choice Question 4

Which operation should be performed before using `sendKeys()` to ensure text replacement works correctly in input fields?

Options:

- A. Execute `clear()` on the element
- B. Call `getText()` to verify the old value
- C. Refresh the browser
- D. Execute an implicit wait command

Skill: Web Element Control

Subskill: Input Field Management

Competency: Ensure proper text overwriting and consistency

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
A. Executing `clear()` empties the input field, ensuring that new text from `sendKeys()` overwrites previous content cleanly.
 - Incorrect Options Feedback:
 - o B. `getText()` does not reset field state.
 - o C. Refresh reloads the page unnecessarily.
 - o D. Implicit wait controls timing, not field content.
-

Multiple Choice Question 5

If `getText()` unexpectedly returns an empty string for a visible field, what is the probable reason?

Options:

- A. The element's text is styled with hidden attributes or resides within an input tag whose value attribute must be accessed instead
- B. There is a network timeout
- C. Browser version mismatch
- D. The element ID is always dynamic

Skill: Automation Debugging

Subskill: Data Extraction

Competency: Diagnose incorrect text retrieval

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Some elements, especially input tags, store content in the `value` attribute rather than visible text nodes, leading to empty `getText()` returns.
 - Incorrect Options Feedback:
 - o B. Timeouts affect availability, not return values.
 - o C. Browser version mismatch would fail earlier.
 - o D. Dynamic IDs affect selection, not extraction.
-

Selenium Waits (Implicit Wait Basics)

Multiple Choice Question 1

What is the main purpose of applying an implicit wait in Selenium tests?

Options:

- A. To pause test execution manually before performing any action
- B. To retry locating elements for a set duration before throwing an exception
- C. To wait for specific conditions like element visibility
- D. To delay page load for debugging

Skill: Synchronization

Subskill: Implicit Wait Fundamentals

Competency: Apply default wait settings to element searches

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
 - B. Implicit waits instruct WebDriver to poll the DOM for a certain time before reporting element absence.
- Incorrect Options Feedback:
 - o A. Manual pauses use `Thread.sleep()`.
 - o C. Conditional checks require explicit waits.
 - o D. Page load timing is browser-controlled.

Multiple Choice Question 2

Which Selenium command correctly configures an implicit wait using the latest Java syntax?

Options:

- A. `driver.setWait(10);`
- B. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));`
- C. `driver.wait(10);`
- D. `WebDriverWait.waitFor(10);`

Skill: WebDriver

Subskill: Wait Configuration

Competency: Use modern Selenium API for wait management

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
 - B. The `implicitlyWait()` method under `manage().timeouts()` sets a global implicit timeout for the session.
 - Incorrect Options Feedback:
 - o A, C, D are not valid Selenium wait methods.
-

Multiple Choice Question 3

What happens if an element appears before the implicit wait timeout expires?

Options:

- A. WebDriver continues without waiting the remaining time
- B. The test fails immediately
- C. WebDriver restarts the timer for other elements
- D. The script pauses until the timeout expires regardless

Skill: Selenium Behavior

Subskill: Implicit Wait Optimization

Competency: Understand dynamic polling behavior

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - A. Implicit wait stops as soon as the element is found, continuing execution instantly.
 - Incorrect Options Feedback:
 - o B. The script passes, not fails.
 - o C. Timers are separate per locator request.
 - o D. Waits terminate early, not always fixed.
-

Multiple Choice Question 4

In which scenario is an implicit wait least effective?

Options:

- A. When handling static elements
- B. When verifying presence of quickly changing AJAX-based content
- C. When waiting for page title updates
- D. When interacting with page headers

Skill: Synchronization

Subskill: Implicit Wait Limitations

Competency: Select appropriate waiting strategy

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - B. Implicit waits only retry element presence, not dynamic states or conditional visibility changes caused by AJAX operations.
 - Incorrect Options Feedback:
 - o A and D. Work fine with static elements.
 - o C. Page titles can be better handled using explicit waits.
-

Multiple Choice Question 5

What effect does setting multiple implicit waits during a session have on execution?

Options:

- A. Multiple waits run simultaneously
- B. Only the last defined wait time value takes effect
- C. All waits are added cumulatively
- D. It causes driver thread conflicts

Skill: Selenium Fundamentals

Subskill: Wait Reconfiguration

Competency: Manage multiple wait definitions responsibly

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - B. Each new implicit wait overrides the previous one, and only the most recent value remains active globally.
 - Incorrect Options Feedback:
 - o A. Selenium manages one active implicit wait at a time.
 - o C. Waits do not accumulate.
 - o D. Timeout redefinition does not cause thread issues.
-

Handling Dynamic Elements

Multiple Choice Question 1

What is the primary challenge when automating dynamic elements in web applications?

Options:

- A. Elements frequently reload their HTML attributes such as ID or class dynamically
- B. Browser drivers fail to launch due to version mismatches
- C. Page titles do not remain consistent
- D. Multiple frames block dynamic element access

Skill: Automation Elements

Subskill: Dynamic Element Behavior

Competency: Recognize synchronization and locator issues

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - A. Dynamic elements often regenerate with changing attribute values, causing locator instability.
- Incorrect Options Feedback:
 - o B. A driver mismatch is an environment configuration issue.
 - o C. Page titles rarely affect locators.
 - o D. Frame handling is a separate concern.

Multiple Choice Question 2

Which XPath function is often used to handle IDs that include numeric strings generated dynamically with each session?

Options:

- A. `starts-with()`
- B. `contains()`
- C. `ends-with()`
- D. `normalize-space()`

Skill: Locator Construction

Subskill: XPath Partial Matching

Competency: Manage dynamic attribute variations

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. The `contains()` function helps match stable portions of dynamically generated attribute values.

- Incorrect Options Feedback:
 - o A. Used when dynamic identifiers start with consistent prefixes.
 - o C. Not natively supported in standard XPath 1.0.
 - o D. Used for trimming spaces, not partial matches.
-

Multiple Choice Question 3

What is the most stable approach to locating frequently changing UI elements?

Options:

- A. Use relative XPath or CSS selectors with static parent relationships
- B. Depend on absolute XPath
- C. Use index-based element selection
- D. Disable dynamic rendering in the browser

Skill: Locator Optimization

Subskill: Dynamic UI Handling

Competency: Apply robust locator strategies

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
 - A. Relative locators stay valid longer because they rely on stable parent-child relationships instead of transient attributes.
 - Incorrect Options Feedback:
 - o B. Absolute paths break upon structure updates.
 - o C. Index-based locators are fragile.
 - o D. Browser rendering cannot simply be disabled.
-

Multiple Choice Question 4

Which Selenium technique helps ensure interaction occurs only after a dynamic element becomes visible?

Options:

- A. Implicit wait
- B. Explicit wait with ExpectedConditions
- C. Immediate click using JavaScriptExecutor
- D. Thread.sleep()

Skill: Synchronization

Subskill: Dynamic Readiness

Competency: Handle dynamic element load timing

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
B. Explicit waits confirm an element's visibility before performing interaction, addressing asynchronous load issues.
 - Incorrect Options Feedback:
 - o A. Implicit waits cannot handle visibility conditions.
 - o C. Forcing JavaScript clicks may bypass visibility verification.
 - o D. Thread.sleep() causes unnecessary delays.
-

Multiple Choice Question 5

When dealing with dynamic drop-down values fetched via AJAX, which method ensures reliable selection?

Options:

- A. Wait for the AJAX call completion with an explicit wait before selecting an option
- B. Preload the HTML options file manually
- C. Refresh the browser each time dropdown changes
- D. Use findElements() with Thread.sleep()

Skill: WebDriver Automation

Subskill: Dynamic Data Synchronization

Competency: Manage AJAX-driven dropdown content

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
A. Waiting until AJAX results populate ensures the dropdown content is available before interaction.
 - Incorrect Options Feedback:
 - o B. Not feasible for production data.
 - o C. Refreshing introduces unnecessary reloads.
 - o D. Fixed delays are unreliable for network latency.
-

Strategies to Locate Dynamic Elements

Multiple Choice Question 1

Which strategy increases locator stability for elements whose IDs are dynamically generated in every session?

Options:

- A. Using relative XPath with a stable parent or ancestor node
- B. Selecting by exact ID match
- C. Using index-based absolute XPath
- D. Injecting static IDs using JavaScript

Skill: Dynamic Locator Strategy

Subskill: XPath Structuring

Competency: Build resilient locator paths for dynamic pages

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
 - A. Relative XPath referencing stable parents provides resilience when child element IDs change dynamically.
 - Incorrect Options Feedback:
 - o B. Exact ID matching fails when IDs change frequently.
 - o C. Absolute paths break with layout changes.
 - o D. JavaScript injection is non-scalable and discouraged.
-

Multiple Choice Question 2

Which CSS selector syntax supports handling dynamic attributes containing predictable substrings?

Options:

- A. `[attribute='staticValue']`
- B. `[attribute*='partialValue']`
- C. `[attribute$='suffixValue']`
- D. `[attribute^='prefixValue']`

Skill: CSS Selectors

Subskill: Substring Matching

Competency: Use attribute pattern selectors effectively

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. The asterisk (*) operator in CSS locators enables partial substring matching, ideal for dynamic identifiers.
 - Incorrect Options Feedback:
 - o A. Matches only static values.
 - o C and D match suffixes and prefixes but not middle patterns.
-

Multiple Choice Question 3

How can `normalize-space()` improve dynamic XPath locators?

Options:

- A. It removes leading and trailing spaces to handle inconsistent text formatting
- B. It filters text based on character encoding
- C. It limits scope to case-sensitive results
- D. It converts dynamic IDs into static strings

Skill: XPath

Subskill: Text Handling

Competency: Manage whitespace issues in dynamic text nodes

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
A. `normalize-space()` ensures text-based XPath expressions remain reliable despite layout or formatting inconsistencies.
 - Incorrect Options Feedback:
 - o B. Not related to encoding.
 - o C. Case sensitivity is unaffected.
 - o D. It doesn't alter attribute values.
-

Multiple Choice Question 4

When working with Angular-based applications where component IDs change dynamically, which approach is more reliable?

Options:

- A. Use attribute-based locators like `ng-model` or `formcontrolname`
- B. Use CSS tag combinations with dynamic IDs
- C. Depend on implicit waits for synchronization
- D. Use `JavaScriptExecutor` for hardcoded element lookup

Skill: Automation Strategy

Subskill: Angular-Specific Element Location

Competency: Implement framework-aware locator techniques

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
A. Angular provides consistent attributes (`ng-model`, `formcontrolname`) ideal for stable locator creation.
 - Incorrect Options Feedback:
 - o B. Dynamic IDs are unstable.
 - o C. Waits do not fix locator volatility.
 - o D. Hardcoding minimizes maintainability.
-

Multiple Choice Question 5

What combination of techniques is most effective for reliably automating web elements with unpredictable hierarchy and identifiers?

Options:

- A. Chain relative XPath expressions with partial text and attribute filters
- B. Use absolute XPath combined with index numbers

- C. Disable Selenium's DOM traversal optimization
- D. Execute random polling with Thread.sleep()

Skill: Automation Design

Subskill: Dynamic Locator Construction

Competency: Create dependable locators in complex UIs

Difficulty Level: Advanced

Bloom Level: Synthesis

- Correct Answer Reason:
 - A. Combining relative paths, attribute conditions, and text snippets enhances adaptability in dynamic DOM structures.
 - Incorrect Options Feedback:
 - o B. Absolute paths are brittle.
 - o C. This setting doesn't exist in Selenium.
 - o D. Random delays reduce efficiency and consistency.
-

Handling Frames and Iframes (switchTo, Nested Frames)

Multiple Choice Question 1

Which Selenium command is used to move the driver's control focus from the main document to an iframe element?

Options:

- A. `driver.navigate().switchToFrame()`
- B. `driver.switchTo().frame()`
- C. `driver.selectFrame()`
- D. `driver.getFrame()`

Skill: Frame Handling

Subskill: Context Switching

Competency: Control frame-level navigation in Selenium

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. The correct syntax `driver.switchTo().frame()` transfers driver control into the specified iframe context.
 - Incorrect Options Feedback:
 - o A, C, D are deprecated or invalid Selenium methods.
-

Multiple Choice Question 2

If a page has nested iframes, how can WebDriver access an element inside the deepest frame?

Options:

- A. Switch directly using frame name of inner frame
- B. Sequentially switch into each nested frame using `driver.switchTo().frame()`
- C. Use implicit waits to automatically access child frames
- D. Refresh the page and reset context

Skill: Frame Navigation

Subskill: Nested Frames

Competency: Manage multi-level frame contexts

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
 - B. WebDriver must switch sequentially into each nested frame before interacting with elements inside deeper frames.
 - Incorrect Options Feedback:
 - o A. Direct access fails without intermediate context.
 - o C. Waits do not control frame navigation.
 - o D. Refresh only resets frame context.
-

Multiple Choice Question 3

Which command reverts WebDriver control from a frame to the main document?

Options:

- A. `driver.switchTo().mainWindow()`
- B. `driver.restoreDefaultContent()`
- C. `driver.switchTo().defaultContent()`
- D. `driver.resetContext()`

Skill: WebDriver Functions

Subskill: Frame Exit

Competency: Manage frame switching lifecycle

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
 - C. The `defaultContent()` command restores driver focus to the main document from any active frame context.
 - Incorrect Options Feedback:
 - o A. `mainWindow()` handles window switching.
 - o B and D are invalid in the WebDriver API.
-

Multiple Choice Question 4

What common issue occurs when interacting with an element inside a frame without switching context first?

Options:

- A. StaleElementReferenceException
- B. NoSuchFrameException
- C. NoSuchElementException
- D. ElementClickInterceptedException

Skill: Frame Troubleshooting

Subskill: Exception Handling

Competency: Identify synchronization errors during frame automation

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason:
C. Selenium throws `NoSuchElementException` when elements are not found due to incorrect frame context.
 - Incorrect Options Feedback:
 - o A. Occurs when a previously found element becomes stale.
 - o B. Triggered if frame name/index is invalid.
 - o D. Happens from overlay interruptions, not frame context.
-

Multiple Choice Question 5

Which best practice ensures maintainable frame handling in complex multi-frame applications?

Options:

- A. Use stable frame identifiers or `WebElement` references with descriptive names
- B. Access frames by index numbers for simplicity
- C. Hardcode frame switch order for every test case
- D. Refresh frame context after each interaction

Skill: Automation Practices

Subskill: Scalable Frame Management

Competency: Apply sustainable design for multi-frame automation

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
A. Referencing frames by unique identifiers or stored `WebElement` variables improves maintainability and reduces hardcoding risks.
 - Incorrect Options Feedback:
 - o B. Index references are easily broken.
 - o C. Hardcoding lacks scalability.
 - o D. Frequent refresh slows execution unnecessarily.
-

Handling Windows and Tabs (getWindowHandles, switchTo)

Multiple Choice Question 1

Which command returns a unique identifier for the currently active browser window in Selenium?

Options:

- A. `driver.getCurrentWindowId()`
- B. `driver.getWindowHandle()`
- C. `driver.switchTo().window()`
- D. `driver.getWindowName()`

Skill: Window Management

Subskill: Handle Identification

Competency: Retrieve and manage multiple browser sessions

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason:
 - B. The `getWindowHandle()` method retrieves the unique identifier (handle) of the active browser window.
 - Incorrect Options Feedback:
 - o A and D do not exist in the Selenium API.
 - o C. Used for switching, not for retrieving the current handle.
-

Multiple Choice Question 2

When multiple browser tabs are open, which Selenium method is used to capture all open window handles at once?

Options:

- A. `driver.getAllWindows()`
- B. `driver.getWindowHandles()`
- C. `driver.switchTo().allHandles()`
- D. `driver.listWindows()`

Skill: Window Navigation

Subskill: Multi-Tab Handling

Competency: Manage and iterate through multiple open sessions

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
 - B. The `getWindowHandles()` method returns a set of window handles corresponding to all active browser instances.
- Incorrect Options Feedback:
 - o A, C, and D are not valid Selenium methods.

Multiple Choice Question 3

After switching to a new window using `driver.switchTo().window(handle)`, what happens to the WebDriver context?

Options:

- A. WebDriver continues executing commands in the previously active window
- B. WebDriver control shifts to the specified browser window
- C. The driver duplicates all open tabs
- D. The original tab automatically closes

Skill: Selenium Control Flow

Subskill: Context Switching

Competency: Manage active browser focus

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
 - B. The driver's execution context transfers to the new window, allowing interaction with elements in that specific window.
- Incorrect Options Feedback:
 - o A. Context changes after switching.
 - o C and D. Selenium does not duplicate or close windows automatically.

Multiple Choice Question 4

What is a recommended approach when dynamically switching between newly opened tabs in Selenium tests?

Options:

- A. Iterate through all handles, compare titles, and switch to the matching one
- B. Assume the last handle is always the newly opened tab
- C. Use random index selection from handles list
- D. Refresh the browser to reset tab order

Skill: Test Automation

Subskill: Dynamic Window Targeting

Competency: Implement efficient tab-switching logic

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason:
 - A. Comparing tab titles or URLs ensures that the correct window or tab receives focus even if handle order varies.
- Incorrect Options Feedback:
 - o B. Handle order is unpredictable.
 - o C. Random selection risks control errors.
 - o D. Refreshing resets the session unnecessarily.

Multiple Choice Question 5

After closing a secondary browser window, which command restores control to the main parent window?

Options:

- A. `driver.switchTo().defaultContent()`
- B. `driver.switchTo().parentFrame()`
- C. `driver.switchTo().window(mainHandle)`
- D. `driver.navigate().backTo(mainHandle)`

Skill: Window Management

Subskill: Context Restoration

Competency: Manage and restore command control

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
C. The handle for the parent window, stored earlier in a variable, can be passed into `switchTo().window()` to regain control.
 - Incorrect Options Feedback:
 - o A. Used for frames, not windows.
 - o B. Applies only to nested iframes.
 - o D. No such navigation method exists.
-

Handling Alerts and Popups (accept, dismiss, getText)

Multiple Choice Question 1

Which Selenium command switches the WebDriver focus from the main page to the active JavaScript alert?

Options:

- A. `driver.switchTo().alert()`
- B. `driver.getAlert()`
- C. `driver.handleAlert()`
- D. `driver.getActiveAlert()`

Skill: Alert Handling

Subskill: Context Switching

Competency: Manage browser alert boxes effectively

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
A. The `switchTo().alert()` command transfers WebDriver's control from the main window to the currently active alert popup.
 - Incorrect Options Feedback:
 - o B, C, and D are invalid methods in Selenium's standard API.
-

Multiple Choice Question 2

What is the result of executing `alert.accept()` in Selenium?

Options:

- A. The alert box is closed by clicking Cancel
- B. The alert box is closed by clicking OK or Accept
- C. It fetches the text of the alert
- D. It dismisses the alert and returns null

Skill: Selenium Commands

Subskill: Alert Interaction

Competency: Perform basic alert operations

Difficulty Level: Intermediate

Bloom Level: Comprehension

- Correct Answer Reason:
B. The `accept()` function simulates the OK or Confirm button click, closing the alert and continuing test execution.
 - Incorrect Options Feedback:
 - o A. `dismiss()` handles cancellation.
 - o C. `getText()` retrieves the alert content.
 - o D. Not applicable to `accept` functionality.
-

Multiple Choice Question 3

If the application displays a confirmation popup and the user wants to click Cancel, which Selenium command should be used?

Options:

- A. `alert.accept()`
- B. `alert.dismiss()`
- C. `alert.exit()`
- D. `alert.ignore()`

Skill: Browser Alerts

Subskill: Interaction Commands

Competency: Execute cancellation operations on confirmation dialogs

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
B. The `dismiss()` method closes the popup by simulating a Cancel button click.
 - Incorrect Options Feedback:
 - o A. Accept performs confirmation.
 - o C and D are not valid Selenium methods.
-

Multiple Choice Question 4

How can text content from an alert popup be extracted in Selenium?

Options:

- A. `alert.getMessage()`
- B. `driver.getAlertText()`
- C. `alert.getText()`
- D. `driver.getPopUpContent()`

Skill: Selenium Interaction

Subskill: Data Retrieval

Competency: Extract alert messages effectively

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason:
C. The `getText()` method retrieves the text displayed on the active alert window for validation or logging.
 - Incorrect Options Feedback:
 - o A. Incorrect method name.
 - o B. Not part of Selenium's API.
 - o D. Selenium doesn't support `getPopUpContent()`.
-

Multiple Choice Question 5

In complex web applications with delayed alert triggering, what approach ensures reliable alert handling without premature failure?

Options:

- A. Use explicit waits to wait until `ExpectedConditions.alertIsPresent()` returns true
- B. Use `Thread.sleep()` for a fixed duration
- C. Continuously poll by reloading the page
- D. Disable alert handling in `WebDriver`

Skill: Synchronization

Subskill: Alert Readiness Handling

Competency: Synchronize script execution with delayed alerts

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason:
A. Explicit waits with `ExpectedConditions.alertIsPresent()` efficiently handle dynamically appearing alerts.
 - Incorrect Options Feedback:
 - o B. Fixed delays reduce accuracy.
 - o C. Refreshing doesn't guarantee alert visibility.
 - o D. Disabling handling prevents automation entirely.
-

That's a deep and structured request, ideal for building a professional Selenium/Java assessment set. Below is a fully formatted set of **five advanced/intermediate MCQs** per topic, following your required structure precisely.

Explicit Waits and ExpectedConditions

Multiple Choice Question 1

Which of the following ExpectedConditions is most appropriate when verifying that a dynamically loaded element is clickable before performing a click action?

Options:

- A. `presenceOfElementLocated()`
- B. `elementToBeClickable()`
- C. `visibilityOfAllElementsLocatedBy()`
- D. `textToBePresentInElement()`

Skill: Selenium waits

Subskill: Applying explicit waits for dynamic elements

Competency: Choosing the correct ExpectedCondition

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `elementToBeClickable()` ensures both visibility and clickability before performing a click, preventing `ElementNotInteractable` errors.
 - Incorrect Options Feedback:
 - o A waits only for presence, not interaction capability.
 - o C applies visibility to multiple elements.
 - o D tests for text content, not visibility or clickability.
-

Multiple Choice Question 2

What happens if the timeout of an explicit wait expires before the condition is met?

Options:

- A. The test is retried automatically.
- B. The test continues without waiting further.

- C. A `TimeoutException` is thrown.
- D. Selenium exits the session.

Skill: Selenium waits

Subskill: Exception handling with waits

Competency: Managing timeout outcomes

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: A `TimeoutException` indicates that the condition was never satisfied within the wait duration.
 - Incorrect Options Feedback:
 - o A is incorrect; Selenium does not retry automatically.
 - o B misinterprets wait behavior.
 - o D is incorrect; sessions persist unless manually ended.
-

Multiple Choice Question 3

To wait for an alert to appear, which `ExpectedCondition` is best?

Options:

- A. `presenceOfElementLocated()`
- B. `alertIsPresent()`
- C. `visibilityOfElementLocated()`
- D. `elementToBeSelected()`

Skill: Selenium waits

Subskill: Handling alerts through waits

Competency: Choosing specific `ExpectedConditions`

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `alertIsPresent()` explicitly waits for alert availability in DOM context.
 - Incorrect Options Feedback:
 - o A and C relate to page elements.
 - o D checks selection states of elements.
-

Multiple Choice Question 4

When should `FluentWait` be preferred over `WebDriverWait`?

Options:

- A. When waiting for a static element
- B. When polling frequency and ignored exceptions must be customized
- C. When using implicit waits only
- D. When managing parallel test execution

Skill: Selenium waits

Subskill: FluentWait configuration

Competency: Customizing polling and exception strategy

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason: FluentWait provides flexibility on polling intervals and exception handling strategies.
 - Incorrect Options Feedback:
 - o A fails to require any dynamic condition.
 - o C and D misrepresent wait usage scope.
-

Multiple Choice Question 5

What is the polling mechanism in WebDriverWait by default?

Options:

- A. Every 100 milliseconds
- B. Every 500 milliseconds
- C. Every 1 second
- D. Every 5 seconds

Skill: Selenium waits

Subskill: Wait mechanics

Competency: Understanding WebDriverWait internals

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: WebDriverWait polls every 500 milliseconds by default.
 - Incorrect Options Feedback:
 - o A is too frequent.
 - o C and D are incorrect defaults.
-

Selenium Advanced Actions

Multiple Choice Question 1

How do you perform multiple combined user actions like click-and-hold followed by moveToElement?

Options:

- A. Using separate driver click() methods
- B. With the Actions class chaining methods then calling perform()
- C. By using JavaScriptExecutor directly
- D. By setting implicit waits before clicking

Skill: Selenium Actions
Subskill: Sequential user gestures
Competency: Implementing composite interactions
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: Actions class builds a complex sequence of gestures that execute on perform().
 - Incorrect Options Feedback:
 - o A separates actions without chaining.
 - o C and D are unrelated approaches.
-

Multiple Choice Question 2

Which action correctly drags an element from source to target?

Options:

- A. dragAndDrop(source, target).build().perform()
- B. moveToElement(target).click().perform()
- C. clickAndHold(target).perform()
- D. contextClick(target).build()

Skill: Selenium Actions
Subskill: Drag-and-drop
Competency: Executing object movement actions
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: dragAndDrop performs the combined hold-and-move-release operation.
 - Incorrect Options Feedback:
 - o B and C are partial actions.
 - o D invokes a right click only.
-

Multiple Choice Question 3

What method moves the mouse pointer to the middle of a web element?

Options:

- A. moveByOffset()
- B. moveToElement()
- C. moveAcrossPage()
- D. scrollToElement()

Skill: Selenium Actions
Subskill: Mouse movement
Competency: Navigating to target element

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: moveToElement() is designed to hover or relocate the cursor to the element center.
 - Incorrect Options Feedback:
 - o A moves by coordinates.
 - o C and D are invalid or non-existent in Actions class.
-

Multiple Choice Question 4

Which of the following enables building a sequence of actions without executing them immediately?

Options:

- A. sendKeys()
- B. build()
- C. performImmediately()
- D. actionChain()

Skill: Selenium Actions

Subskill: Action sequencing

Competency: Managing deferred execution

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason: build() compiles queued actions for later use, executed via perform().
 - Incorrect Options Feedback:
 - o A executes instantly.
 - o C and D are not valid Selenium methods.
-

Multiple Choice Question 5

For sending keyboard inputs to a non-focused element, which Action is appropriate?

Options:

- A. keyDown(Keys.TAB)
- B. sendKeys(element, "text")
- C. click().sendKeys("text")
- D. keyUp(Keys.CONTROL)

Skill: Selenium Actions

Subskill: Keyboard simulation

Competency: Controlling exposed element interactions

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason: `sendKeys(element, "text")` sends input directly to a given element.
 - Incorrect Options Feedback:
 - o A, C, D involve modifier keys or unintended actions.
-

Excellent. Here is the complete continuation for topics **3 to 14**, following the same instructional and structural design standards.

Dropdowns (`Select` class: single and multi-select)

Multiple Choice Question 1

How can a single-select dropdown option be chosen by visible text?

Options:

- A. `select.selectByText("Option")`
- B. `select.selectByVisibleText("Option")`
- C. `driver.click("Option")`
- D. `select.selectByDisplayName("Option")`

Skill: Dropdown handling

Subskill: Selection by text

Competency: Using Select class methods correctly

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `selectByVisibleText()` matches an option's visible text directly as displayed.
 - Incorrect Options Feedback:
 - o A uses an incorrect method name.
 - o C bypasses the Select class.
 - o D references an invalid method.
-

Multiple Choice Question 2

How do you verify if a dropdown allows multiple selections?

Options:

- A. `select.isMultiple()`
- B. `select.isMultiSelectable()`
- C. `driver.isMultiple()`
- D. `select.hasMultiple()`

Skill: Dropdown handling

Subskill: Multi-select capability

Competency: Checking dropdown type

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: `isMultiple()` checks whether more than one selection is possible.
 - Incorrect Options Feedback:
 - B, C, D are invalid in the Select class API.
-

Multiple Choice Question 3

Which method clears all selected options in a multi-select dropdown?

Options:

- A. `deselectAll()`
- B. `selectAll()`
- C. `clearSelections()`
- D. `removeOptions()`

Skill: Dropdown handling

Subskill: Clearing selections

Competency: Controlling selection state

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `deselectAll()` resets all active option states.
 - Incorrect Options Feedback:
 - B adds selections instead.
 - C and D do not exist in the API.
-

Multiple Choice Question 4

To retrieve all available options in a dropdown, which method should be used?

Options:

- A. `getAllElements()`
- B. `getOptions()`
- C. `getSelectables()`
- D. `getList()`

Skill: Dropdown handling

Subskill: Extracting dropdown contents

Competency: Iterating over dropdown elements

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `getOptions()` returns a list of `WebElements` contained in the dropdown.

- Incorrect Options Feedback:
 - o A, C, D are not valid Select class methods.
-

Multiple Choice Question 5

Which of the following statements correctly selects the third option by index?

Options:

- A. `select.selectByIndex(3)`
- B. `select.selectByIndex(2)`
- C. `select.chooseByIndex(3)`
- D. `select.selectAt(2)`

Skill: Dropdown handling

Subskill: Selection by index

Competency: Understanding index-based selection

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: Indexing in Selenium starts from 0, so the third option is `selectByIndex(2)`.
 - Incorrect Options Feedback:
 - o A selects the fourth option.
 - o C and D use invalid methods.
-

Checkboxes and Radio Button Automation

Multiple Choice Question 1

How do you confirm whether a checkbox is selected?

Options:

- A. `checkbox.isDisplayed()`
- B. `checkbox.isSelected()`
- C. `checkbox.isEnabled()`
- D. `checkbox.getText().equals("checked")`

Skill: Element state management

Subskill: Checkbox validation

Competency: Verifying selection state

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `isSelected()` identifies whether the checkbox or radio button is active.
- Incorrect Options Feedback:
 - o A checks visibility.

- o C validates usability.
 - o D compares unrelated text properties.
-

Multiple Choice Question 2

What happens when clicking a radio button within a group?

Options:

- A. All buttons become selected.
- B. The previously selected option is automatically deselected.
- C. The browser throws an exception.
- D. Multiple buttons can be selected simultaneously.

Skill: Form automation

Subskill: Radio group logic

Competency: Understanding mutual exclusivity

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: Only one radio option per group remains selected.
 - Incorrect Options Feedback:
 - o A, C, D contradict HTML behavior.
-

Multiple Choice Question 3

How would you programmatically select a checkbox only if it's not already checked?

Options:

- A. `if(checkbox.isSelected()) checkbox.click();`
- B. `if(!checkbox.isSelected()) checkbox.click();`
- C. `checkbox.click()` always;
- D. `clickOnlyIfUnchecked(checkbox);`

Skill: Conditional logic in actions

Subskill: Checkbox toggling

Competency: Preventing redundant actions

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: The code checks for unchecked state before clicking.
 - Incorrect Options Feedback:
 - o A deselects when already selected.
 - o C lacks condition.
 - o D is not valid syntax.
-

Multiple Choice Question 4

Which locator strategy is best for selecting a particular radio button from a group?

Options:

- A. Use `findElements()` and iterate matching value attribute.
- B. Select all radio elements using tag name.
- C. Use `findElementByText()` on label.
- D. Rely on document index ordering.

Skill: Locator design

Subskill: Attribute targeting

Competency: Selecting elements safely

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason: Iterating through `findElements()` by value ensures accurate selection.
 - Incorrect Options Feedback:
 - o B is too broad.
 - o C unreliable unless label is linked explicitly.
 - o D is fragile.
-

Multiple Choice Question 5

If a checkbox is hidden using CSS, how can it be checked through automation?

Options:

- A. `checkbox.click()` directly
- B. Use `JavaScriptExecutor` to set `checked=true`
- C. `driver.executeScript("click()")` without locating element
- D. Use `moveToElement()` and `click()`

Skill: Hidden element handling

Subskill: JavaScript-based actions

Competency: Interacting with concealed inputs

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason: `JavaScriptExecutor` directly updates properties even for non-visible elements.
 - Incorrect Options Feedback:
 - o A causes `ElementNotInteractableException`.
 - o C lacks proper reference.
 - o D requires visible coordinates.
-

Mouse and Keyboard Interactions (Actions class: hover, drag-and-drop, double-click)

Multiple Choice Question 1

How do you perform a hover over a specific element?

Options:

- A. `moveToElement(element).perform()`
- B. `hoverOver(element).build()`
- C. `scrollToElement(element)`
- D. `click(element).perform()`

Skill: Actions API

Subskill: Mouse hover

Competency: Creating precise cursor positioning

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `moveToElement` performs hovering actions in Selenium.
 - Incorrect Options Feedback:
 - o B invalid API method.
 - o C and D execute other actions.
-

Multiple Choice Question 2

What method double-clicks an element?

Options:

- A. `doubleClick(element).perform()`
- B. `clickTwice(element)`
- C. `multiClick(element,2)`
- D. `repeatClick(element)`

Skill: Actions API

Subskill: Double-click gesture

Competency: Using proper API for dual clicks

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `doubleClick(element)` triggers two consecutive clicks in one command.
 - Incorrect Options Feedback:
 - o B–D are non-existing methods.
-

Multiple Choice Question 3

What is the primary difference between `dragAndDrop` and `dragAndDropBy`?

Options:

- A. `dragAndDropBy` uses coordinate offset while `dragAndDrop` uses element targets.
- B. `dragAndDropBy` works only with images.
- C. `dragAndDrop` is asynchronous.
- D. No difference.

Skill: Mouse Actions

Subskill: Drag accuracy

Competency: Selecting suitable movement type

Difficulty Level: Advanced

Bloom Level: Analysis

- Correct Answer Reason: `dragAndDropBy` works on relative coordinates.
 - Incorrect Options Feedback:
 - o B–D misstate API function distinctions.
-

Multiple Choice Question 4

Which key combination can be simulated with the Actions class?

Options:

- A. `keyDown(Keys.CONTROL).sendKeys("A").keyUp(Keys.CONTROL).perform()`
- B. `sendKeys(Keys.SHIFT, "A")`
- C. `keyHold("CTRL+A")`
- D. none; not supported

Skill: Keyboard Actions

Subskill: Shortcut key simulation

Competency: Simulating complex keyboard inputs

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason: The combination chain simulates Ctrl+A correctly.
 - Incorrect Options Feedback:
 - o B–C not native methods.
 - o D is false.
-

Multiple Choice Question 5

How can you perform sequential key input followed by mouse click?

Options:

- A. `sendKeys("text").click().perform()`
- B. `action.sendKeys("text").perform().click()`
- C. `build().click().perform()`
- D. `executeComboKeys("text", click())`

Skill: Action chaining
Subskill: Sequential gestures
Competency: Chain-building for compound interactions
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: Chaining sendKeys and click before perform executes actions chronologically.
 - Incorrect Options Feedback:
 - o B executes in separate steps.
 - o C incomplete.
 - o D invalid.
-

JavaScriptExecutor (scrolling, clicking, handling hidden elements)

Multiple Choice Question 1

Which method executes a JavaScript click on an element?

Options:

- A. executeScript("arguments.click();", element)
- B. runScript("element.click()")
- C. performScriptClick(element)
- D. element.click()

Skill: JavaScriptExecutor
Subskill: Executing scripts
Competency: Handling hidden or unresponsive elements
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: executeScript executes JavaScript directly, interacting even with hidden elements.
 - Incorrect Options Feedback:
 - o B–C are invalid methods.
 - o D uses the native WebDriver click, failing on hidden items.
-

Multiple Choice Question 2

To scroll until an element is in view using JavaScript, which command is correct?

Options:

- A. executeScript("arguments.scrollToView(true);", element)
- B. executeScript("window.scrollTo(0,window.length)")

- C. `executeScript("element.scroll();")`
- D. `executeScript("scrollToBottom();")`

Skill: `JavaScriptExecutor`

Subskill: Scrolling actions

Competency: Script-triggered viewport adjustment

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `scrollIntoView(true)` brings the element into visible viewport area.
 - Incorrect Options Feedback:
 - o B scrolls randomly.
 - o C incorrect context.
 - o D assumes non-existent function.
-

Multiple Choice Question 3

Which data type does `executeScript()` return?

Options:

- A. Always String
- B. Object depending on the script result
- C. Boolean only
- D. `WebDriver` instance

Skill: `JavaScriptExecutor`

Subskill: Script execution return values

Competency: Handling dynamic data returns

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason: The method returns Object depending on script result type (e.g., String, Boolean, `WebElement`).
 - Incorrect Options Feedback:
 - o A–C are partially limited results.
 - o D incorrect context.
-

Multiple Choice Question 4

When using `JavaScriptExecutor` in Selenium, what interface must the driver implement?

Options:

- A. `WebDriver`
- B. `JavascriptExecutable`
- C. `JavascriptExecutor`
- D. `RemoteWebInterface`

Skill: JavaScriptExecutor
Subskill: Interface identification
Competency: Recognizing driver capabilities
Difficulty Level: Intermediate
Bloom Level: Knowledge

- Correct Answer Reason: Drivers like ChromeDriver inherently implement JavascriptExecutor.
 - Incorrect Options Feedback:
 - o A too general.
 - o B and D invalid interface names.
-

Multiple Choice Question 5

Which command retrieves the page title using JavaScriptExecutor?

Options:

- A. `executeScript("return document.title;")`
- B. `executeScript("document.title();")`
- C. `executeScript("window.getTitle;")`
- D. `executeScript("document.retrieveTitle")`

Skill: JavaScriptExecutor
Subskill: DOM property accessing
Competency: Retrieving properties with JavaScript
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: The return statement fetches title directly.
 - Incorrect Options Feedback:
 - o B incorrect syntax.
 - o C and D invalid property references.
-

Automating End-to-End Form Submission with Validations

Multiple Choice Question 1

What is the best sequence while automating a form submission?

Options:

- A. Locate fields → Fill values → Validate → Submit
- B. Submit → Fill → Validate
- C. Validate → Fill → Locate → Submit
- D. Fill → Submit → Locate fields

Skill: Automation flow
Subskill: Sequential logic
Competency: Structuring automation tasks logically
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: Fields must be located and filled before validations and submission.
 - Incorrect Options Feedback:
 - o B–D disrupt logical order.
-

Multiple Choice Question 2

To verify client-side validation messages, which WebDriver method helps?

Options:

- A. `getAttribute("validationMessage")`
- B. `getText()`
- C. `checkValidation()`
- D. `getErrorMessage()`

Skill: Form validation
Subskill: Attribute use
Competency: Capturing client-side responses
Difficulty Level: Advanced
Bloom Level: Application

- Correct Answer Reason: `getAttribute("validationMessage")` extracts the browser's built-in error text.
 - Incorrect Options Feedback:
 - o B retrieves visible text only.
 - o C and D unsupported.
-

Multiple Choice Question 3

Which strategy ensures clean test data before new form submissions?

Options:

- A. Use `driver.navigate().refresh()`
- B. Pre-fill form with dummy invalid data
- C. `driver.quit()` before every test
- D. Reset values using `form.clear()`

Skill: Test preparation
Subskill: Data management
Competency: Maintaining state consistency
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: refresh ensures the form reloads with pristine state.
 - Incorrect Options Feedback:
 - o B not clearing data.
 - o C too extreme.
 - o D invalid method.
-

Multiple Choice Question 4

To handle dynamic confirmation messages after form submission, which locator approach is stable?

Options:

- A. CSS or XPath with text() contains()
- B. Hard-coded id
- C. Index-based element
- D. XPath using position()

Skill: Dynamic element handling

Subskill: Text partial match

Competency: Locating transient elements

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason: contains() handles dynamic text changes effectively.
 - Incorrect Options Feedback:
 - o B–D brittle or static.
-

Multiple Choice Question 5

Why should validation checks be included after form submission?

Options:

- A. To confirm backend and frontend integration works
- B. To shorten runtime
- C. To test mouse movement
- D. To bypass manual entry

Skill: Validation logic

Subskill: End-to-end assurance

Competency: Confirming correctness of results

Difficulty Level: Intermediate

Bloom Level: Evaluation

- Correct Answer Reason: End-to-end validation guarantees correct workflow completion.
- Incorrect Options Feedback:
 - o B–D irrelevant.

TestNG Basics and Page Object Model (POM)

Multiple Choice Question 1

Which POM purpose aligns with test modularity?

Options:

- A. Keeping locators and tests in one file
- B. Separating page interactions from test logic
- C. Writing everything in main()
- D. Hardcoding element locators

Skill: Framework design

Subskill: POM structure

Competency: Maintainable test architecture

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason: POM improves reusability by isolating interaction logic.
- Incorrect Options Feedback:
 - o A, C, D contradict principles.

Multiple Choice Question 2

What is TestNG's main purpose in a Selenium project?

Options:

- A. Managing browser installation
- B. Organizing and executing tests with annotations
- C. Handling CSS locators only
- D. Creating data files

Skill: TestNG fundamentals

Subskill: Framework usage

Competency: Using TestNG effectively

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: TestNG provides annotation-based structured test management.
- Incorrect Options Feedback:
 - o A, C, D incorrect responsibilities.

Multiple Choice Question 3

Which POM advantage directly impacts maintainability?

Options:

- A. Code duplication
- B. Centralized change control for locators
- C. Random data injection
- D. Test case repetition

Skill: POM structure

Subskill: Maintenance advantage

Competency: Applying design consistency

Difficulty Level: Intermediate

Bloom Level: Evaluation

- Correct Answer Reason: Centralizing locator definitions enables easy maintenance.
 - Incorrect Options Feedback:
 - o A, C, D negative aspects.
-

Multiple Choice Question 4

What does @Test annotation in TestNG signify?

Options:

- A. A configuration file
- B. A runnable test method
- C. A utility function
- D. A driver instantiation block

Skill: Test organization

Subskill: Test definition

Competency: Using annotations

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: @Test marks a method to be executed as a test.
 - Incorrect Options Feedback:
 - o A–D misinterpret roles.
-

Multiple Choice Question 5

Where should page actions like clickLogin() reside in POM?

Options:

- A. Inside test case classes
- B. In the page class representing that page
- C. In a configuration XML
- D. Inline in data provider

Skill: POM implementation

Subskill: Action distribution

Competency: Correct separation of page functions

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: Page classes encapsulate the operations performed on elements they model.
 - Incorrect Options Feedback:
 - o A repeats logic.
 - o C and D unrelated layers.
-

TestNG Annotations (@Test, @BeforeMethod, @AfterMethod, etc.)

Multiple Choice Question 1

What is the execution order for methods annotated with @BeforeMethod and @AfterMethod?

Options:

- A. Both executed before the test
- B. @BeforeMethod before and @AfterMethod after each @Test
- C. Both run once after all tests
- D. Both never run automatically

Skill: TestNG lifecycle

Subskill: Annotation behavior

Competency: Understanding execution flow

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: @BeforeMethod initializes setup per test, while @AfterMethod handles cleanup.
 - Incorrect Options Feedback:
 - o A, C, D incorrect lifecycle states.
-

Multiple Choice Question 2

Which annotation runs only once before all test methods in a class?

Options:

- A. @BeforeTest
- B. @BeforeClass
- C. @BeforeMethod
- D. @BeforeSuite

Skill: TestNG lifecycle

Subskill: Class-level setup

Competency: Optimizing environment setup

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `@BeforeClass` executes once prior to all `@Test` methods within that class.
 - Incorrect Options Feedback:
 - A, C, D different scopes.
-

Multiple Choice Question 3

If multiple `@Test` methods have no priority defined, in what order do they execute?

Options:

- A. Alphabetically by method name
- B. Randomly
- C. As declared in file
- D. By system time

Skill: TestNG configuration

Subskill: Execution ordering

Competency: Predicting behavior

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: In absence of priority, TestNG runs methods alphabetically.
 - Incorrect Options Feedback:
 - B–D incorrect.
-

Multiple Choice Question 4

Which annotation executes even if a test fails?

Options:

- A. `@AfterMethod`
- B. `@BeforeSuite`
- C. `@BeforeMethod`
- D. `@AfterTest`

Skill: Cleanup automation

Subskill: Postcondition execution

Competency: Ensuring resource closure

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `@AfterMethod` always runs regardless of test result.
- Incorrect Options Feedback:
 - B–D have different scopes.

Multiple Choice Question 5

Which is true about @BeforeSuite?

Options:

- A. Executes after every test
- B. Runs before the first test suite begins
- C. Runs once after all suite tests
- D. Executes per class

Skill: TestNG configuration

Subskill: Suite-level events

Competency: Handling suite setup

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: @BeforeSuite executes once before any testing activity begins.
- Incorrect Options Feedback:
 - o A, C, D improper timing.

Assertions in TestNG (Hard vs Soft)

Multiple Choice Question 1

What is the main difference between hard and soft assertions?

Options:

- A. Hard assertions continue even after failure
- B. Soft assertions allow continuation after failure
- C. Soft assertions stop immediately
- D. No functional difference

Skill: Assertion handling

Subskill: Error control

Competency: Selecting assertion types

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason: Soft assertions allow remaining validations to execute.
- Incorrect Options Feedback:
 - o A, C, D incorrect understanding.

Multiple Choice Question 2

Which class is used for soft assertions in TestNG?

Options:

- A. AssertSoft
- B. SoftAssert
- C. SoftValidation
- D. ValidationSoft

Skill: Assertions

Subskill: Class usage

Competency: Implementing non-blocking checks

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: SoftAssert handles deferred assertion verification.
 - Incorrect Options Feedback:
 - o A, C, D invalid class names.
-

Multiple Choice Question 3

What must be invoked to finalize soft assertions?

Options:

- A. assertAll()
- B. finishAll()
- C. close()
- D. completeSoft()

Skill: Assertions

Subskill: Verification

Competency: Collating results

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: assertAll() throws final evaluation error, marking fails.
 - Incorrect Options Feedback:
 - o B–D invalid methods.
-

Multiple Choice Question 4

Why are hard assertions used in smoke tests?

Options:

- A. They gather all errors
- B. They halt execution at first issue
- C. They ignore failures
- D. They delay failure reporting

Skill: Assertion logic

Subskill: Test control

Competency: Reliable quick validation

Difficulty Level: Intermediate

Bloom Level: Evaluation

- Correct Answer Reason: Immediate termination is ideal for primary validation.
 - Incorrect Options Feedback:
 - o A, C, D opposite behavior.
-

Multiple Choice Question 5

What happens when assertEquals fails in hard assertion?

Options:

- A. Test continues
- B. Execution stops for that test
- C. Exception is ignored
- D. Suite quits entirely

Skill: Assertion behavior

Subskill: Hard failure handling

Competency: Understanding exceptions

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: AssertionError stops further code in that test method.
 - Incorrect Options Feedback:
 - o A, C, D incorrect flow.
-

TestNG Test Suites (`testng.xml`), Grouping, and Execution

Multiple Choice Question 1

What is the primary use of `testng.xml` in a project?

Options:

- A. To declare environment variables
- B. To configure and organize test execution hierarchy
- C. To define browser capabilities
- D. To compile test classes into JAR files

Skill: TestNG suite configuration

Subskill: Suite management

Competency: Understanding XML-driven test organization

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: `testng.xml` allows controlling test order, grouping, and parallel execution.
 - Incorrect Options Feedback:
 - o A and C pertain to other files.
 - o D is unrelated to TestNG configuration.
-

Multiple Choice Question 2

Which tag inside `testng.xml` defines a logical group of test cases?

Options:

- A. `<packages>`
- B. `<test>`
- C. `<group>`
- D. `<suite>`

Skill: TestNG XML structure

Subskill: Tag identification

Competency: Defining grouping levels

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: `<test>` logically groups multiple classes or methods.
 - Incorrect Options Feedback:
 - o A includes packages, not groups.
 - o C invalid tag for structure.
 - o D denotes the top-level suite container.
-

Multiple Choice Question 3

Which annotation allows adding methods to a named group?

Options:

- A. `@Group("name")`
- B. `@Test(groups="groupName")`
- C. `@Include(group="name")`
- D. `@RunGroup("groupName")`

Skill: Test grouping

Subskill: Annotation configuration

Competency: Associating tests into categories

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: The groups parameter in `@Test` assigns methods to test groups.
- Incorrect Options Feedback:
 - o A, C, D invalid syntax.

Multiple Choice Question 4

Which attribute in `testng.xml` enables running specific groups only?

Options:

- A. included-groups
- B. group-run
- C. include-tests
- D. group-select

Skill: Suite execution

Subskill: Controlled group inclusion

Competency: Executing subset of tests

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason: included-groups selects specific categories from `@Test` groups.
- Incorrect Options Feedback:
 - o B–D invalid TestNG attributes.

Multiple Choice Question 5

How can tests run in parallel across classes?

Options:

- A. Set `parallel="classes"` and `thread-count` in suite configuration
- B. Use `@ParallelTest` annotation
- C. Duplicate the same tests in multiple files
- D. Thread management is unsupported

Skill: Parallel execution

Subskill: Performance optimization

Competency: Configuring concurrency

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason: The `parallel="classes"` attribute allows simultaneous class execution.
 - Incorrect Options Feedback:
 - o B invalid annotation.
 - o C redundant.
 - o D false.
-

Page Object Model (POM): Principles and Implementation

Multiple Choice Question 1

What is the main design intent of POM?

Options:

- A. Merge data and test code
- B. Separate page structure from test scripts
- C. Write all locators inline
- D. Store elements in random classes

Skill: POM fundamentals

Subskill: Design architecture

Competency: Understanding page abstraction

Difficulty Level: Intermediate

Bloom Level: Knowledge

- Correct Answer Reason: POM achieves separation of responsibilities, improving readability.
 - Incorrect Options Feedback:
 - o A, C, D contradict modular design.
-

Multiple Choice Question 2

What elements does a POM class typically include?

Options:

- A. Only locators
- B. Locators and methods performing actions
- C. Entire test cases
- D. Test data only

Skill: POM structure

Subskill: Element composition

Competency: Writing page classes

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: POM classes combine locators with methods representing user actions.
 - Incorrect Options Feedback:
 - o A minimal scope.
 - o C and D incorrect responsibilities.
-

Multiple Choice Question 3

Why is POM beneficial for large teams?

Options:

- A. It ensures only one team member writes code
- B. It centralizes locators for easy maintenance
- C. It increases testing difficulty
- D. It forces test logic within pages

Skill: Collaboration and maintainability

Subskill: Central management

Competency: Enhancing teamwork in automation

Difficulty Level: Intermediate

Bloom Level: Evaluation

- Correct Answer Reason: Centralized maintenance reduces locator duplication.
 - Incorrect Options Feedback:
 - o A, C, D incorrect.
-

Multiple Choice Question 4

Which annotation helps initialize elements in POM?

Options:

- A. `@FindBy`
- B. `@ElementLocate`
- C. `@Locator`
- D. `@FindElement`

Skill: POM Implementation

Subskill: PageFactory annotation

Competency: Automatic element initialization

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: `@FindBy` identifies element location strategies for PageFactory.
 - Incorrect Options Feedback:
 - o B–D are invalid.
-

Multiple Choice Question 5

What is the main advantage of using PageFactory?

Options:

- A. Lazy element initialization for performance
- B. Immediate driver quitting

- C. Manual driver injection
- D. Random page creation

Skill: POM architecture

Subskill: Initialization advantage

Competency: Efficient instance management

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason: PageFactory loads elements on-demand, enhancing performance.
 - Incorrect Options Feedback:
 - o B–D irrelevant.
-

Separating Locators and Actions in POM

Multiple Choice Question 1

Why should locators be separated from actions in POM?

Options:

- A. To duplicate selectors
- B. To improve maintainability when UI changes
- C. To mix logic and UI
- D. To confuse page design

Skill: Modular design

Subskill: Locator isolation

Competency: Achieving cleaner architecture

Difficulty Level: Intermediate

Bloom Level: Analysis

- Correct Answer Reason: Separating locators ensures changes propagate easily without affecting logic.
 - Incorrect Options Feedback:
 - o A, C, D contradict architecture best practices.
-

Multiple Choice Question 2

Where should element locators ideally be defined?

Options:

- A. In test methods
- B. Inside page class or a dedicated locator file
- C. In configuration properties
- D. In data providers

Skill: POM discipline
Subskill: File organization
Competency: File-level organization consistency
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: Page class encapsulates locators or inherits from a locator repository.
 - Incorrect Options Feedback:
 - o A, C, D misplace visual object identifiers.
-

Multiple Choice Question 3

How should page methods interact with locators?

Options:

- A. Interact directly via `driver.findElement()` in tests
- B. Use encapsulated interactions in page methods calling the locators internally
- C. Retrieve locators from properties files manually
- D. Combine locators and actions inline

Skill: Page encapsulation
Subskill: Interaction abstraction
Competency: Writing reusable action methods
Difficulty Level: Intermediate
Bloom Level: Application

- Correct Answer Reason: Encapsulating `findElement` calls inside the page class ensures abstraction.
 - Incorrect Options Feedback:
 - o A and D reduce modular efficiency.
 - o C add unnecessary complexity.
-

Multiple Choice Question 4

What is a result of poor separation between locators and actions?

Options:

- A. Easier debugging
- B. Fragile scripts with redundant maintenance
- C. Reduced complexity
- D. Improved scalability

Skill: Architecture evaluation
Subskill: Design integrity
Competency: Recognizing architectural flaws
Difficulty Level: Intermediate
Bloom Level: Evaluation

- Correct Answer Reason: Tight coupling leads to redundancy and high maintenance.
 - Incorrect Options Feedback:
 - A, C, D contradict result of poor design.
-

Multiple Choice Question 5

How can you externalize locators effectively?

Options:

- A. Use property or JSON files read at runtime
- B. Hardcode them in multiple classes
- C. Store locators in database
- D. Avoid using locators

Skill: Locator management

Subskill: Externalization strategies

Competency: Scalable locator maintenance

Difficulty Level: Advanced

Bloom Level: Application

- Correct Answer Reason: Externalized locator files enable centralized updates and flexibility.
 - Incorrect Options Feedback:
 - B–D impractical or unsupported.
-

Implementing POM in a Selenium Project

Multiple Choice Question 1

What is the first step when implementing POM?

Options:

- A. Create page classes for each application page
- B. Write all tests in one file
- C. Initialize driver globally only
- D. Ignore object encapsulation

Skill: Implementation planning

Subskill: Starting structure

Competency: Initial project setup

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: Each page class mirrors one application page for structured automation.
- Incorrect Options Feedback:
 - B–D contradict design principles.

Multiple Choice Question 2

How is reusability achieved in POM?

Options:

- A. By repeating locators per test
- B. By centralizing actions in page classes
- C. By writing inline driver calls in test methods
- D. By using static tests only

Skill: Reusability

Subskill: Code reuse

Competency: Sharing page logic

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: Centralizing logic avoids duplication across tests.
- Incorrect Options Feedback:
 - o A, C, D incorrect.

Multiple Choice Question 3

What best integrates POM with TestNG?

Options:

- A. Instantiate page objects within `@BeforeMethod` setup
- B. Create new driver instance per method
- C. Store driver inside XML file
- D. Avoid `@BeforeMethod` altogether

Skill: Framework integration

Subskill: Object lifecycle management

Competency: Properly initializing test prerequisites

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: Creating page objects in setup ensures fresh state for each test.
- Incorrect Options Feedback:
 - o B mismanages driver lifecycle.
 - o C–D invalid implementation.

Multiple Choice Question 4

What ensures scalability in a POM-based project?

Options:

- A. Single combined class
- B. Modular reusable components for each page and utility
- C. No directory structure
- D. Randomized test flows

Skill: Scalability

Subskill: System design

Competency: Extending automation easily

Difficulty Level: Advanced

Bloom Level: Evaluation

- Correct Answer Reason: Modular organization allows multiple contributors and new page additions effortlessly.
 - Incorrect Options Feedback:
 - o A, C, D oppose modular goals.
-

Multiple Choice Question 5

How do you link tests with actions in POM?

Options:

- A. By importing corresponding page object classes and calling their methods
- B. By copying action code into test files
- C. By running driver commands manually from tests
- D. By saving actions in Excel sheets

Skill: Test integration

Subskill: Test-action connection

Competency: Implementing test-to-page communication

Difficulty Level: Intermediate

Bloom Level: Application

- Correct Answer Reason: The test class imports page classes and triggers high-level page actions.
 - Incorrect Options Feedback:
 - o B, C, D inefficient or invalid.
-