**1. Login Form Automation**

* **WebDriver Methods:** findElement, sendKeys, click, getTitle, getCurrentUrl
* **Scenario:** Automate the login process on a demo site. Validate that the user is redirected to the correct page after successful login.
* **Challenge:** Locate input fields, handle incorrect login, and capture login error messages.

**2. Navigation and Verification**

* **WebDriver Methods:** navigate().to, navigate().back, navigate().forward, navigate().refresh
* **Scenario:** Open a website, navigate to different sections, then use the back and forward navigation options to test if the history navigation works as expected.
* **Challenge:** Verify page titles at each navigation step and refresh the page at the end.

**3. Handling Dropdowns and Radio Buttons**

* **WebDriver Methods:** findElement, click, isSelected, Select class for dropdowns
* **Scenario:** Interact with a form that contains dropdowns, checkboxes, and radio buttons. Select values and verify that the right options are selected.
* **Challenge:** Practice selecting options by index, visible text, and value for dropdowns. Ensure the right selection is made for radio buttons and checkboxes.

**4. File Upload**

* **WebDriver Methods:** sendKeys (for file path), click
* **Scenario:** Automate file upload by sending the file path to an input of type file on a demo site.
* **Challenge:** Verify successful upload by checking for any confirmation message or uploaded file preview.

**5. Form Filling and Validation**

* **WebDriver Methods:** clear, sendKeys, submit
* **Scenario:** Fill out a lengthy form with various input types. Automate clearing, inputting text, and submitting the form.
* **Challenge:** Check form validation messages by leaving some required fields empty.

**6. Alert Handling**

* **WebDriver Methods:** switchTo().alert(), accept, dismiss, getText
* **Scenario:** Trigger alerts, capture their text, and perform actions (accept or dismiss).
* **Challenge:** Verify that the alert handling works correctly and affects the page as expected.

**7. Window and Tab Switching**

* **WebDriver Methods:** getWindowHandles, switchTo().window
* **Scenario:** Open multiple tabs/windows and switch between them.
* **Challenge:** Ensure that each window/tab is handled separately and close only specific ones at the end of the test.

**8. Scrolling**

* **WebDriver Methods:** executeScript for JavaScript-based scrolling
* **Scenario:** Scroll down to a specific element on a page or to the bottom.
* **Challenge:** Validate that specific elements become visible after scrolling.

**9. Frames Handling**

* **WebDriver Methods:** switchTo().frame, switchTo().defaultContent
* **Scenario:** Switch between frames on a page to interact with elements inside each frame.
* **Challenge:** Locate elements inside different frames and switch back to the main content.

**10. Handling Dynamic Elements**

* **WebDriver Methods:** implicitlyWait, explicitlyWait, WebDriverWait, ExpectedConditions
* **Scenario:** Interact with elements that take time to load (e.g., loading spinners or AJAX-based content).
* **Challenge:** Apply waits effectively to ensure stable automation without hardcoded delays.

These scenarios cover a range of WebDriver functions and are designed to build your understanding of how each function behaves in different contexts. Let me know if you’d like help setting up code for any of these scenarios

1. **1. Handling Nested Iframes and Alerts**
   * **WebDriver Methods:** switchTo().frame, switchTo().alert(), switchTo().parentFrame()
   * **Scenario:** Automate interactions within multiple nested iframes, where one of the frames triggers a JavaScript alert.
   * **Challenge:** Identify which iframe to switch to, handle alerts without breaking the sequence, and verify that the interactions only apply within the specified frames. Switch back to the main content accurately after all interactions.

**2. Dynamic Dropdowns and Auto-Suggestions**

* + **WebDriver Methods:** sendKeys, findElements, Actions class
  + **Scenario:** Type into a search field with auto-suggestions, wait for the suggestions list to load, and select a specific option.
  + **Challenge:** Handle lists that load dynamically and may vary by search term. Choose an option by text without relying on static indexes, and verify the suggestion selection.

**3. Dealing with Stale Elements in a Loading Loop**

* + **WebDriver Methods:** explicitlyWait, ExpectedConditions.refreshed, try-catch blocks
  + **Scenario:** Interact with elements on a page where content keeps refreshing or updating, causing StaleElementReferenceException.
  + **Challenge:** Use waits to ensure you interact only when elements are stable, or implement a retry mechanism that catches stale element exceptions and retries the action after a brief delay.

**4. Invisible or Hidden Elements Handling**

* + **WebDriver Methods:** executeScript (JavaScript), isDisplayed, click
  + **Scenario:** Interact with elements that are present in the DOM but hidden by default (like a menu that appears on hover).
  + **Challenge:** Use JavaScript to make the elements visible, or hover over a specific area to trigger the visibility. Confirm that your actions apply only to visible elements.

**5. Managing CAPTCHA with Alternative Solutions**

* + **Scenario:** Encounter a CAPTCHA on login but still automate as much of the flow as possible.
  + **Challenge:** Automate until the CAPTCHA, then handle scenarios for manual CAPTCHA solving (e.g., add waits or manual prompts). Alternatively, practice handling CAPTCHA bypasses if your organization provides them for automation testing purposes.

**6. Testing a SPA (Single-Page Application) with Dynamic Elements**

* + **WebDriver Methods:** JavaScriptExecutor for AJAX loads, waits
  + **Scenario:** Automate interactions on an SPA where elements are frequently updated without page reloads, such as clicking tabs or loading new sections.
  + **Challenge:** Use JavaScriptExecutor to monitor for changes in elements or for AJAX calls to complete. Validate the appearance of elements only after full data loads, and avoid false positives from incomplete updates.

**7. Multiple Windows or Pop-ups with Random Identifiers**

* + **WebDriver Methods:** getWindowHandles, switchTo().window, close
  + **Scenario:** A pop-up with a random identifier opens after clicking a button. Automate switching to the new window, perform actions, and close it.
  + **Challenge:** Identify which window is the new one when multiple windows are open, interact with it, then switch back to the original window and confirm no interactions affect other open windows.

**8. Handling Drag-and-Drop with Elements Changing Size or Location**

* + **WebDriver Methods:** Actions.dragAndDrop, executeScript (for custom drag-and-drop)
  + **Scenario:** Perform drag-and-drop actions on elements that are dynamically sized or repositioned after dropping.
  + **Challenge:** Implement error handling for cases where the element has moved or changed size, and verify the accuracy of the final position. Consider using JavaScript drag-and-drop in cases where WebDriver’s Actions class is inconsistent.

**9. Filling Forms with Conditional Logic (Multi-Step Form)**

* + **WebDriver Methods:** findElement, click, sendKeys, explicitlyWait
  + **Scenario:** Automate a form that shows additional fields based on earlier selections.
  + **Challenge:** Detect and handle fields that are dynamically displayed based on previous inputs. Use waits to handle conditionally displayed fields, ensure only relevant fields are interacted with, and test multiple conditions.

**10. Complex Table Interactions with Pagination**

* + **WebDriver Methods:** findElements, getText, click
  + **Scenario:** Interact with a table that spans multiple pages and has specific rows based on conditions (e.g., select rows with values above a threshold).
  + **Challenge:** Navigate across pages to locate specific rows, handle scenarios where rows may change position due to sorting or filtering, and perform actions only on identified rows.

These scenarios challenge your ability to use WebDriver methods effectively in unpredictable and complex environments. They help strengthen debugging, error handling, and adaptive scripting skills, which are vital for robust automation. Let me know if you'd like code snippets or further breakdowns on any specific scenario!

Great question! The scenarios above primarily focus on WebDriver and some interactions with WebElements, but if you want to practice WebElement-specific methods in more detail, here are specific scenarios that target them directly:

**Key WebElement Methods and Scenarios to Practice**

* + getText() and getAttribute()
    - **Scenario:** Automate a product details page where product information (e.g., name, price, description) is displayed. Extract text content using getText() and verify that it matches expected values.
    - **Challenge:** Use getAttribute() to fetch hidden attributes like data-\* attributes, value for input fields, or href for links to ensure the page is set up correctly.
  + isDisplayed()**,**isEnabled()**, and**isSelected()
    - **Scenario:** Automate a multi-step form with checkboxes, radio buttons, and disabled input fields based on certain conditions. Confirm visibility, enabled state, and selection state of elements before interacting.
    - **Challenge:** Use conditional statements to verify each element’s state and handle scenarios where elements may be unexpectedly disabled or hidden.
  + clear()**and**sendKeys()
    - **Scenario:** Automate filling a search field on an e-commerce site. Type a query, clear it, and type a new one to ensure the clear() method works effectively.
    - **Challenge:** Verify that the field is cleared completely before typing a new query and ensure special characters can be entered without issues.
  + click()
    - **Scenario:** Interact with a menu that has multiple nested options. Use click() to navigate through each option, expanding submenus and verifying the correct items appear.
    - **Challenge:** Handle cases where clicking a parent element may toggle visibility of its children or may unexpectedly close the entire menu. Ensure clicks are applied only when elements are ready.
  + getCssValue()
    - **Scenario:** Verify visual elements, such as checking the background color of a button or font color of text after performing specific actions.
    - **Challenge:** Check that style changes (like color or font) are applied correctly based on interactions, such as hovering, clicking, or form validation.
  + getSize()**and**getLocation()
    - **Scenario:** Validate UI layout by measuring elements’ sizes and positions. For instance, confirm that buttons are aligned or that images maintain a certain dimension on the page.
    - **Challenge:** Verify the position or size of elements on responsive pages, where dimensions might change based on screen size.
  + submit()
    - **Scenario:** Automate a form submission by directly calling submit() on an input element.
    - **Challenge:** Ensure that form validation messages appear correctly and confirm submission success without manually clicking a submit button.

**Combining WebElement Methods in Complex Scenarios**

* 1. **Form Validation with Clear and Attribute Checking**
     + **Scenario:** Automate a login form where incorrect entries cause fields to highlight in red and display error messages. Use getCssValue() to confirm field color changes and getAttribute() to validate error messages in aria-\* attributes.
     + **Challenge:** Combine clear(), sendKeys(), and click() to test multiple invalid inputs, confirming proper validation responses.
  2. **Dynamic Element Styling and State Checking**
     + **Scenario:** Automate a toggle switch that changes states and styles (like color or position). Use click(), isEnabled(), and getCssValue() to confirm state and style changes.
     + **Challenge:** Validate the toggle's behavior when it’s interacted with in different states and use getText() or getAttribute() to capture any state labels.
  3. **Extracting List Information Using**findElements()**and**getText()
     + **Scenario:** Automate a news site’s homepage, extract article titles and links from a list, and validate that each link is unique and points to a valid article.
     + **Challenge:** Use findElements() to retrieve multiple elements and iterate over them with getText() and getAttribute(), verifying the content structure and link formatting.

These scenarios specifically emphasize WebElement methods and provide practice with key interactions that involve validation, state management, and styling. They build a solid foundation for mastering element-level interactions in Selenium WebDriver. Let me know if you need detailed examples for any method!

**questions** reported by candidates who have **recently interviewed at Infosys** for automation roles, specifically for **5 years of experience**. These questions have been asked in real interviews for automation testing positions at Infosys:

**1. String Manipulation:**

* **Check if a string is a palindrome.**
* **Reverse a string without using built-in functions.**
* **Find the longest substring without repeating characters.**
* **Count the number of vowels and consonants in a given string.**

**2. Array and List Manipulation:**

* **Find the second largest element in an array.**
* **Find the missing number in an array of 1 to N (where one number is missing).**
* **Write a program to sort an array of strings in lexicographical order.**
* **Move all zeroes to the end of an array without changing the relative order of non-zero elements.**

**\* multiplication of no except for the current index**

**3. Linked List:**

* **Reverse a linked list.**
* **Find the middle element of a linked list.**
* **Detect a cycle in a linked list (Floyd’s cycle-finding algorithm).**
* **Remove duplicates from a linked list.**

**4. Sorting and Searching:**

* **Write a program to implement bubble sort.**
* **Write a program to implement selection sort.**
* **Write a program to perform binary search on a sorted array.**
* **Implement quicksort or mergesort.**

**5. Numbers and Math Problems:**

* **Check if a number is a prime number.**
* **Find the factorial of a number using recursion.**
* **Print Fibonacci series up to a given number.**
* **Find the greatest common divisor (GCD) of two numbers.**
* **Write a program to find Armstrong numbers in a range.**

**6. OOP and Design Patterns:**

* **What is the difference between method overloading and method overriding?**
* **Create a Singleton class in Java.**
* **Write a Java program demonstrating the concept of abstraction using abstract classes.**
* **Write a program to demonstrate the use of inheritance and polymorphism.**

**7. Data Structures and Algorithms:**

* **Implement a stack using arrays/linked lists.**
* **Implement a queue using two stacks.**
* **Find the shortest path in a graph (using Dijkstra’s algorithm).**
* **Implement depth-first search (DFS) and breadth-first search (BFS) for a graph.**

**8. Selenium and Automation Testing:**

* **Write a Selenium script to open a browser, navigate to a website, and click on a button.**
* **Write a Selenium WebDriver script to capture a screenshot of a page.**
* **How would you handle dynamic elements in Selenium (e.g., elements whose attributes change frequently)?**
* **Write a Selenium script to perform file upload using the sendKeys() method.**
* **Write a Selenium script to switch between different browser windows.**
* **How would you handle a scenario where a web element appears after waiting for a certain period (AJAX calls)?**

**9. TestNG and Framework Questions:**

* **What is the difference between @BeforeTest and @BeforeSuite in TestNG?**
* **How do you run multiple tests in parallel in TestNG?**
* **Write a TestNG script to read data from a CSV file and use it in test cases.**
* **How would you implement a logging framework for test execution in your automation project?**
* **Explain how you would create a Page Object Model (POM) design pattern.**

**10. Database Testing:**

* **Write a SQL query to find duplicate records in a table.**
* **Write a query to find the second highest salary in a table.**
* **How would you validate data between the UI and the database in automation testing?**

**11. Scenario-Based Coding:**

* **Write a program to find the intersection of two arrays.**
* **Write a program to find the largest rectangular area in a histogram.**
* **Write a program to merge two sorted arrays.**
* **Write a program to calculate the power of a number using recursion.**

**12. Miscellaneous/General Questions:**

* **Write a Java program to implement the Fibonacci series using both recursion and iteration.**
* **Implement a basic calculator in Java that supports addition, subtraction, multiplication, and division.**
* **How would you find the missing number in an array containing numbers from 1 to N?**

**13. System Design (Advanced Level for Frameworks):**

* **Design a generic automation framework for web-based applications (including reporting, logging, and exception handling).**
* **How would you design a hybrid framework that combines data-driven and keyword-driven approaches?**
* **How do you handle dynamic web elements in your framework design?**