Selenium WebDriver Basic Questions

1. **What is Selenium?** Selenium is an open-source suite of tools used for automating web browsers. It supports multiple programming languages (Java, Python, C#, etc.) and web browsers, making it a popular choice for automating web applications for testing purposes.
2. **Difference between Selenium 3 and Selenium 4?**
   - **Selenium 3:** Limited to using the legacy RemoteWebDriver, lacks support for modern browser features like better W3C WebDriver support.
   - **Selenium 4:** Fully supports the W3C WebDriver standard, improved grid functionality, better browser interaction APIs, and better handling of modern web standards.
3. **What are the different types of locators in Selenium?**
   - By.id()
   - By.name()
   - By.className()
   - By.linkText()
   - By.partialLinkText()
   - By.tagName()
   - By.xpath()
   - By.cssSelector()
4. **Difference between findElement() and findElements()?**
   - findElement(): Returns the first matching element or throws a NoSuchElementException if no match is found.
   - findElements(): Returns a list of matching elements. If no match is found, it returns an empty list.
5. **How do you handle dropdowns in Selenium?** Use the Select class to interact with dropdowns. Example:

   ```
   Select dropdown = new Select(driver.findElement(By.id("dropdownId")));

   dropdown.selectByVisibleText("Option 1");
   ```
6. **How to handle multiple windows in Selenium?** Use driver.getWindowHandles() to get all window handles and switch between them using driver.switchTo().window(handle).
7. **What is the difference between get() and navigate().to()?**
   - get(): Opens a URL and waits for the page to load completely.
   - navigate().to(): Also opens a URL but does not necessarily wait for the page to fully load.

8. **How do you handle alerts and pop-ups?** Use the Alert interface to handle alerts:

   Alert alert = driver.switchTo().alert();

   alert.accept(); // To accept the alert

   alert.dismiss(); // To dismiss the alert

9. **How do you handle frames and iframes?** Use driver.switchTo().frame() to switch to a frame. You can pass an index, name, or WebElement:

   driver.switchTo().frame("frameName");

10. **What is the difference between driver.quit() and driver.close()?**
    - quit(): Closes all browser windows and ends the WebDriver session.
    - close(): Closes the current browser window, but the WebDriver session may remain active.

Selenium WebDriver Advanced Questions

11. **What are the different types of waits in Selenium?**
    - **Implicit Wait:** Waits for a set amount of time before throwing a NoSuchElementException.
    - **Explicit Wait:** Waits for a specific condition to occur before proceeding.
    - **Fluent Wait:** A more flexible wait, where you can specify polling frequency and ignore specific exceptions.
12. **Implicit vs Explicit Wait - What's the difference?**
    - **Implicit Wait:** Applies globally to all elements for a specified amount of time.
    - **Explicit Wait:** Used for specific elements with a condition that must be met (e.g., element visibility, element clickability).
13. **What is Fluent Wait, and when do you use it?** Fluent Wait is a type of wait that allows you to specify the frequency with which the condition is checked and also allows ignoring specific exceptions like NoSuchElementException. It's typically used for complex waiting scenarios.
14. **How do you handle 'StaleElement ReferenceException'?** It happens when an element is no longer attached to the DOM. To handle it, re-find the element after waiting for the element to become available again.
15. **What is JavaScriptExecutor? How do you use it?** JavaScriptExecutor allows you to execute JavaScript code within the context of the browser. You can use it to perform actions that aren't supported by WebDriver directly.

    JavascriptExecutor js = (JavascriptExecutor) driver;

    js.executeScript("window.scrollBy(0,1000)");

16. **How do you scroll a webpage using Selenium?** You can use JavaScriptExecutor to scroll the page:

    ```
    JavascriptExecutor js = (JavascriptExecutor) driver;

    js.executeScript("window.scrollBy(0,500)");
    ```

17. **How do you take a screenshot in Selenium?** Use TakesScreenshot interface to capture screenshots:

    ```
    File src = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

    FileUtils.copyFile(src, new File("path_to_save_image"));
    ```

18. **How do you handle file uploads in Selenium?** Selenium can't directly handle file uploads in a dialog, but you can interact with the file input element (usually an <input type="file">):

    ```
    driver.findElement(By.id("fileUpload")).sendKeys("C:\\path\\to\\file");
    ```

19. **How do you validate broken links in Selenium?** Use HttpURLConnection to check the response status code for all links:

    ```
    URL url = new URL("https://example.com");

    HttpURLConnection connection = (HttpURLConnection) url.openConnection();

    connection.setRequestMethod("HEAD");

    connection.connect();

    int responseCode = connection.getResponseCode();
    ```

20. **How do you capture network logs in Selenium?** You can use the LoggingPreferences class to capture network logs (mostly in Chrome or Firefox):

    ```
    LoggingPreferences logs = new LoggingPreferences();

    logs.enable(LogEntries.Type.PERFORMANCE, Level.ALL);

    ChromeOptions options = new ChromeOptions();

    options.setCapability("goog:loggingPrefs", logs);
    ```

Selenium Framework Questions for Best Practices

21. **What is the Page Object Model (POM)?** POM is a design pattern in Selenium where each web page is represented by a separate class. This helps in organizing and maintaining code better.

22. **What is Page Factory? How is it different from POM?** Page Factory is a part of POM that helps initialize elements with annotations like @FindBy. It's an improved version of POM where you don't need to write manual code for element initialization.

23. **What is the difference between @FindBy and driver.findElement()?**

    - @FindBy: It's an annotation used in Page Factory to locate elements at runtime.

- driver.findElement(): A method to find an element directly using locators like By.id(), By.xpath(), etc.

24. **What are Selenium Grid and its advantages?** Selenium Grid allows you to run tests in parallel across multiple machines or browsers. It helps speed up test execution and is useful for distributed testing.
25. **How do you handle dynamic elements in Selenium?** Use dynamic locators like XPath with contains(), starts-with(), or CSS selectors to handle elements with dynamic attributes.
26. **What is the role of Desired Capabilities in Selenium?** Desired Capabilities allow you to set specific properties of the browser, like browser version, OS, or other configurations, which are necessary for cross-browser testing.
27. **What are the different types of Assertions used in Selenium?**
    - **Assert**: Hard assertion, will stop the test if the condition fails.
    - **SoftAssert**: Allows the test to continue even if an assertion fails.
28. **What are the limitations of Selenium WebDriver?**
    - Can't handle alerts outside of the browser (like OS-level popups).
    - Can't handle CAPTCHA.
    - Limited support for handling multiple browser tabs in some cases.
29. **How do you integrate Selenium with TestNG?** You can use TestNG to run your Selenium tests. Define test methods with @Test, and use TestNG annotations like @BeforeMethod, @AfterMethod, etc., for setup and teardown.
30. **What are TestNG Listeners, and how do you implement them?** Listeners in TestNG allow you to perform actions before and after certain events, such as tests starting or ending. Implement interfaces like ITestListener to create custom listeners.

Selenium Execution & Debugging Questions

31. **How do you run Selenium tests in headless mode?** You can run tests without a UI using a headless browser configuration in Chrome or Firefox:

    ```
    ChromeOptions options = new ChromeOptions();

    options.addArguments("--headless");

    WebDriver driver = new ChromeDriver(options);
    ```
32. **How do you handle authentication pop-ups in Selenium?** You can pass credentials in the URL or use AutoIT/Robot class for handling authentication pop-ups.
33. **How to execute parallel tests in Selenium?** Use TestNG's parallel attribute in the XML configuration to run tests in parallel across multiple browsers or threads.

34. **How do you handle CAPTCHA in Selenium?** CAPTCHA can't be handled directly, but you can use services like 2Captcha or reCAPTCHA solving tools for bypassing CAPTCHA challenges during tests.

35. **What are the different ways to maximize a browser?**

    - Using driver.manage().window().maximize().
    - Setting specific dimensions using driver.manage().window().setSize(new Dimension(width, height)).