SELENIUM

Advantages of Selenium:

- Open-source and free.
- Cross-browser testing support.
- Supports multiple programming languages.
- Cross-platform compatibility (Windows, macOS, Linux).
- Integration with DevOps tools (Jenkins, Maven, etc.).
- Parallel test execution (Selenium Grid).
- Active community support.
- Efficient automation of web applications.

Disadvantages of Selenium:

- Limited to web applications only.
- No built-in reporting.
- Requires programming knowledge.
- High maintenance for dynamic web elements.
- No official technical support.
- No native support for image-based testing.
- Difficult to automate Captcha and OTPs.
- Limited control over the browser for system-level operations.

Locators

1. ID: By.id("element_id")
2. Name: By.name("element_name")
3. Class Name: By.className("class_name")
4. Tag Name: By.tagName("tag_name")
5. Link Text: By.linkText("link_text")
6. Partial Link Text: By.partialLinkText("partial_link_text")
7. CSS Selector: By.cssSelector("css_selector")
8. XPath: By.xpath("xpath")

Basic XPath :

1. Select by Tag Name: //tagname
2. Select by Attribute: //tagname[@attribute='value']
3. Select by Text Content: //tagname[text()='textValue']

4. Contains Attribute Value: //tagname[contains(@attribute, 'partialValue')]
5. Contains Text Value: //tagname[contains(text(),'partialText')]
6. Select by Class Name: //tagname[@class='classname']
7. Select nth Child (Indexing): //tagname[index]
8. Select by Any Node: //*
1. Select Last Element: //tagname[last()]
2. Select with normalize-space(): //tagname[contains(normalize- space(),'text')]

- **Methods of SearchContext-:**
  - findElement(), FindElements(By.arg)
- **Methods of WebDriver-:**
  - Close(), get(), getCurrentUrl(), getPageSource(), getTitle(), getWindowHandle(), getWindowHandles(), manage(), navigate(), quit(), switchTo()
- **Methods of JavaScriptExecutor-:**
  - executeAsyncScript(), executeScript()
- **Method of TakesScreenshot-:**
  - getScreenshotAs()
- **Method of WebElement-:**
  Clear(), click(), getAttribute(), getCssValue(), getLocation(), getRect(), getSize(), getTagName(), getText(), isDisplayed(), isEnabled(), isSelected(), sendkeys(), submit()

Q. JavaScript to perform scrolling actions

Scroll down by pixels:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
 js.executeScript("window.scrollBy(0,1000)"); // Scroll down by 1000
```

Scroll to an element:

```
WebElement element = driver.findElement(By.id("element_id"));
    JavascriptExecutor js = (JavascriptExecutor) driver;
    js.executeScript("arguments[0].scrollIntoView(true);", element);
```

Without using sendkeys how do you send an input to a text field.

Using Selenium, we can execute JavaScript code to set a value in an input field.

```
// Assuming driver is a valid WebDriver instance and elem is the WebElement
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("arguments[0].value='your input text';", elem);
```

Q. How do you handle alerts, windows, and frames in Selenium?

- I handle alerts using the Alert interface.
- Switch between windows using getWindowHandles() and switchTo().window()
- switch between frames using switchTo().frame()

Q. How do you handle Alerts in Selenium?

- Using switchTo().alert()
- handle it using accept(), dismiss(), getText(), or sendKeys().

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

- **Accepting an alert:** alert.accept();
- **Dismissing an alert:** alert.dismiss();

Getting alert text:

```
String alertText = alert.getText();
System.out.println("Alert text: " + alertText);
```

Sending text to a prompt alert:

```
alert.sendKeys("Your text here");
alert.accept();
```

Q. How do you handle multiple windows in Selenium WebDriver?

- Using getWindowHandles() to get all window IDs,
- Then switch using switchTo().window(windowID).

Q. Reading & Writing Data from Excel File and Property File

```
public void readDataFromExcelFile(String sheetName, int rowNo, int
columnNo,String FILE_PATH) {
    FileInputStream fis = new FileInputStream(FILE_PATH);
    Workbook wb = WorkbookFactory.create(fis);
    wb.getSheet(sheetName).getRow(rowNo).getCell(columnNo).getStringCellValue
    ();
```

```
public void writeDataExcelFile(String sheetName, int rowNo, int columnNo, String
writeValue,String FILE_PATH)  {
        FileInputStream fis = new FileInputStream(FILE_PATH);
        Workbook wb = WorkbookFactory.create(fis);
        Sheet sh = wb.getSheet(sheetName);
        sh.getRow(rowNo).getCell(columnNo).setCellValue(writeValue);
        FileOutputStream fos = new FileOutputStream(FILE_PATH);
        wb.write(fos); wb.close();}
```

```
public void readPropertyData(String key, String FILE_PATH) throws
IOException {
            FileInputStream fis=new FileInputStream(FILE_PATH);
            Properties prop=new Properties();
            prop.load(fis);
            prop.getProperty(key);
```

**************File Upload**************

```
public void selectFileToUpload (WebElement fileInputElement) {
            File file = new File("PATH of jpg/img");
            String absolutePath = file.getAbsolutePath();
            fileInputElement.sendKeys(absolutePath);}
```

Q. How to take screenshot

```
        TakesScreenshot ts = (TakesScreenshot) driver;
        File source = ts.getScreenshotAs (OutputType.FILE);
        File destination = new File (System.getProperty("user.dir") +
"/ScreenShot/");
        FileUtils.copyFile(source, destination);
```

Q. Explain the concept of Waits in Selenium.

1. **Implicit Wait**: Waits a set time for elements to appear everywhere.
2. **Explicit Wait**: Waits for a specific condition before continuing.
3. **Fluent Wait**: Like Explicit Wait, but checks the condition repeatedly until a timeout.

implicitlyWait -:
driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(10));

Explicit Wait Condition

WebDriverWait wait = new WebDriverWait(*driver*,
Duration.*ofSeconds*(120));

Element is visible:

1. WebElement element =
   wait.until(ExpectedConditions.**visibilityOfElementLocated**(locator);
2. wait.until(ExpectedConditions.**presenceOfElementLocated**(locator)
3. wait.until(ExpectedConditions.**elementToBeClickable**(locator)
4. Boolean titleContains =
   wait.until(ExpectedConditions.**titleContains**("partial_title")
5. Boolean titleIs = wait.until(ExpectedConditions.**titleIs**("exact_title"));
6. Boolean textPresent =
   wait.until(ExpectedConditions.**textToBePresentInElementLocated**
   (locato r, "expected_text"));
7. Boolean elementSelected =
   wait.until(ExpectedConditions.**elementToBeSelected**(By.id("elemen
   t_id")));
8. Alert alert = wait.until(ExpectedConditions.**alertIsPresent**());

Q. list of methods from the Select class in Selenium:

```
Select select = new Select(element);

select.selectByValue(value);
```

1. selectByVisibleText(String text)
2. selectByIndex(int index)
3. selectByValue(String value)
4. deselectByVisibleText(String text)
5. deselectByIndex(int index): (only for mul ti-select).
6. deselectByValue(String value): (only for multi-select).
7. deselectAll(): Deselects all options (only for multi-select).
8. getOptions(): Returns a list of all options in the dropdown.
9. getAllSelectedOptions(): (only for multi-select).
10. getFirstSelectedOption(): Returns the first selected option in the dropdown.

Q. Create an instance of Actions class

```
Actions actions = new Actions(driver);
```

1. **click action**

```
WebElement element = driver.findElement(By.id("exampleId"));
actions.click(element).perform();
```

2. **drag and drop action**

```
WebElement source = driver.findElement(By.id("sourceId"));
WebElement target = driver.findElement(By.id("targetId"));
actions.dragAndDrop(source, target).perform();
```

3. **moving to an element**

```
actions.moveToElement(element).perform();
```

4. actions.**doubleClick**(element).perform();

5. **Right Click to an element**

```
actions.contextClick(element).perform();
```

6. actions.**scrollToElement**(element).perform();

Q. TestNG annotations in Selenium

1. @BeforeSuite -: Runs before the entire test suite.
2. @BeforeTest -: Runs before any test method in the <test> tag.
3. @BeforeClass-: Runs before the first method of the current class.
4. @BeforeMethod-: Runs before each test method.
5. @Test-: Marks a method as a test method.
6. @AfterMethod-: Runs after each test method.
7. @AfterClass-: Runs after all the methods in the current class.
8. @AfterTest-: Runs after all the test methods in the <test> tag.
9. @AfterSuite-: Runs after the entire test suite.
10. @DataProvider-: It allows passing multiple sets of data to a test method. Useful for data-driven testing.

Q. Assertion -:

- Assertion is a feature present in TestNG which is used to verify the actual and expected result of the test script.
- As per role of automation every expected result should be verify with assert statement instead of Java if-else statement, because if-else block does not have capacity to fail the test script.

In assertion there are 2 types-: Assert (Hard Assert) and Soft assert

assetEquals(), assertNotEquals(), assertSame(), assertNotSame(), assertNull(), assertNotNull(), assertTrue(), assertFalse(), fail().

Q. Difference Assert and Soft Assert?

Assert

- All the methods are static
- If the comparison fails remaining statements will not be executed in the current method.
- We do not call AssertAll()

SoftAssert

- All the methods are not-static.
- Executes remaining statements even if comparison fails.

Q. Common Exceptions in Selenium WebDriver:

- **NoSuchElementException**: Thrown when an element is not found in the DOM.
- **ElementNotVisibleException**: Thrown when an element is present in the DOM but not visible. (DOM- Document Object Model)
- **ElementNotInteractableException**: Thrown when an element is present and visible but not interactable.
- **ElementClickInterceptedException**: Thrown when an element that is supposed to be clicked is obscured by another element.
- **TimeoutException**: Thrown when a command does not complete in the given time.
- **StaleElementReferenceException**: Thrown when a reference to an element is stale (i.e., the element is no longer present in the DOM).
- **WebDriverException**: A general exception that can be thrown by any WebDriver command.
- **InvalidArgumentException**: Thrown when an invalid argument is passed to a method.

Q. Batch Execution & Group Execution:

- **Batch execution** allows us to run multiple test classes or methods together in a single run
- This is useful when we want to execute a set of tests as a group, either sequentially or in parallel
- we can define these batches in a testng.xml file

Group Execution:

- Group execution lets us categorize test methods into different groups. we can run specific groups of tests based on our needs
- This is particularly useful for separating tests by functionality, priority, or any other criteria
- Groups can be defined using the @Test annotation with the groups attribute

Q. How do you handle dynamic elements in Selenium?

- To handle dynamic elements, I use Explicit Waits with ExpectedConditions to wait for elements to be present, visible, or clickable.
- I also use CSS selectors or XPath to locate elements based on partial attributes.

Q. What is the difference between findElement and findElements?

1. findElement returns a single WebElement, and throws a NoSuchElementException if the element is not found.
2. findElements returns a list of WebElements and an empty list if no elements are found.

Q. What is the Page Factory in Selenium?

- It's a class in Selenium that helps initialize web elements defined in Page Objects, using @FindBy annotation.
- It improves code readability and makes maintenance easier.