Selenium Interview Questions / bullet points

WebDriver supports  HtmlUnitDriver which is a GUI less or headless browser.

WebDriver uses the browser's native compatibility to automation

WebDriver / Selenium RC supports dynamic finders

WebDriver supports the implementation of Listeners

WebDriver communicates directly with the web browsers. Thus making it much faster.

Different Types of locators

* ID
* ClassName
* Name
* TagName
* LinkText
* PartialLinkText
* Xpath
* CSS Selector
* DOM

Xpath can be absolute path(/)  and relative path(//)

**What is Same origin policy and how it can be handled?**

Origin is a sequential combination of scheme, host, and port of the URL. For example, for a URL https://www.softwaretestinghelp.com/resources/, **the origin is a combination of http, softwaretestinghelp.com, 80 correspondingly**.

Thus the Selenium Core (JavaScript Program) **cannot access the elements from an origin that is different from where it was launched**. For Example, if I have launched the JavaScript Program from “https://www.softwaretestinghelp.com”, then I would be able to access the pages within the same domain such as “https://www.softwaretestinghelp.com/resources” or “https://www.softwaretestinghelp.com/istqb-free-updates/”. The other domains like google.com, seleniumhq.org would no more be accessible.

**Q #18) How do I launch the browser using WebDriver?**

The following syntax can be used to launch Browser:  
WebDriver driver = **new** FirefoxDriver();  
WebDriver driver = **new** ChromeDriver();  
WebDriver driver = **new** InternetExplorerDriver();

Implicitly wait:

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

Explicit wait:

WebDriverWait wait=new WebDriverWait(driver, 20);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( “”))).click();

wait.until(ExpectedConditions.*alertIsPresent*());

**Have Many methods in ExpectedConditions:**

1. alertIsPresent()
2. elementSelectionStateToBe()
3. elementToBeClickable()
4. elementToBeSelected()
5. frameToBeAvaliableAndSwitchToIt()
6. invisibilityOfTheElementLocated()
7. invisibilityOfElementWithText()
8. presenceOfAllElementsLocatedBy()
9. presenceOfElementLocated()
10. textToBePresentInElement()
11. textToBePresentInElementLocated()
12. textToBePresentInElementValue()
13. titleIs()
14. titleContains()
15. visibilityOf()
16. visibilityOfAllElements()
17. visibilityOfAllElementsLocatedBy()
18. visibilityOfElementLocated()

**How can you find if an element in displayed on the screen?**

WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

1. isDisplayed()
2. isSelected()
3. isEnabled()

**Syntax:**

**isDisplayed():**  
***boolean****buttonPresence = driver.findElement(By.id(“gbqfba”)).isDisplayed();*

**isSelected():**  
***boolean****buttonSelected = driver.findElement(By.id(“gbqfba”)).isSelected();*

**isEnabled():**  
***boolean****searchIconEnabled = driver.findElement(By.id(“gbqfb”)).isEnabled();*

**Q #24) How to select value in a dropdown?**

The value in the dropdown can be selected using WebDriver’s Select class.

**Syntax:**

**selectByValue:**  
*Select selectByValue =****new****Select(driver.findElement(By.id(“SelectID\_One”)));*  
*selectByValue.selectByValue(“greenvalue”);*

**selectByVisibleText:**  
*Select selectByVisibleText =****new****Select (driver.findElement(By.id(“SelectID\_Two”)));*  
*selectByVisibleText.selectByVisibleText(“Lime”);*

**selectByIndex:**  
*Select selectByIndex =****new****Select(driver.findElement(By.id(“SelectID\_Three”)));*  
*selectByIndex.selectByIndex(2);*

**Q #27)** **How to**[**handle frame in WebDriver**](https://www.softwaretestinghelp.com/selenium-tutorial-18/)**?**

An inline frame acronym as iframe is used to insert another document within the current HTML document or simply a web page into a web page by enabling nesting.

**Select iframe by id**  
*driver.switchTo().frame(“ID of the frame“);*

**Locating iframe using tagName**  
*driver.switchTo().frame(driver.findElements(By.tagName(“iframe”).get(0));*

**Locating iframe using index**

**frame(index)**  
*driver.switchTo().frame(0);*

**frame(Name of Frame)**  
*driver.switchTo().frame(“name of the frame”);*

**frame(WebElement element)**  
**Select Parent Window**  
*driver.switchTo().defaultContent();*

**Q #29)** **How to find more than one web element in the list?**

// Storing the list

List <WebElement> elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li"));

// Fetching the size of the list

int listSize = elementList.size();

for (int i=0; i<listSize; i++)

{

// Clicking on each service provider link

serviceProviderLinks.get(i).click();

// Navigating back to the previous page that stores link to service providers

driver.navigate().back();

}

**Q #31) Can Selenium handle windows based pop up?**

Selenium is an automation testing tool which supports only web application testing. Therefore, windows pop up cannot be handled using Selenium.

**Q #32) How can we handle web-based pop-up?**

WebDriver offers the users a very efficient way to [handle these pop-ups using Alert interface](https://www.softwaretestinghelp.com/handle-alerts-popups-selenium-webdriver-selenium-tutorial-16/). There are the four methods that we would be using along with the Alert interface.

* void dismiss() – The dismiss() method clicks on the “Cancel” button as soon as the pop-up window appears.
* void accept() – The accept() method clicks on the “Ok” button as soon as the pop-up window appears.
* String getText() – The getText() method returns the text displayed on the alert box.
* void sendKeys(String stringToSend) – The sendKeys() method enters the specified string pattern into the alert box.

**Syntax:**  
*// accepting javascript alert*  
*Alert alert = driver.switchTo().alert();*  
*alert.accept();*

**Q #35) How to mouse hover on a web element using WebDriver?**

**Sample Code:**

|  |
| --- |
| // Instantiating Action Interface  Actions actions=new Actions(driver);  // howering on the dropdown  actions.moveToElement(driver.findElement(By.id("id of the dropdown"))).perform();  // Clicking on one of the items in the list options  WebElement subLinkOption=driver.findElement(By.id("id of the sub link"));  subLinkOption.click(); |

**Q #37) How to capture screenshot in WebDriver?**

|  |
| --- |
| import org.junit.After;  import org.junit.Before;  import org.junit.Test;  import java.io.File;  import java.io.IOException;  import org.apache.commons.io.FileUtils;  import org.openqa.selenium.OutputType;  import org.openqa.selenium.TakesScreenshot;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;    public class CaptureScreenshot {  WebDriver driver;  @Before  public void setUp() throws Exception {  driver = new FirefoxDriver();  driver.get("https://google.com");  }  @After  public void tearDown() throws Exception {  driver.quit();  }    @Test  public void test() throws IOException {  // Code to capture the screenshot  File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);  // Code to copy the screenshot in the desired location  FileUtils.copyFile(scrFile, new File("C:\\CaptureScreenshot\\google.jpg"))  }  } |

**Q #44) What are the different types of frameworks?**

**Below are the different types of frameworks:**

1. **Module Based Testing Framework:** The framework divides the entire “Application Under Test” into the number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts have taken together builds a larger test script representing more than one module.
2. **Library Architecture Testing Framework:** The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.
3. Data Driven Testing Framework: Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The data is conventionally stored in “Key-Value” pairs. Thus, the key can be used to access and populate the data within the test scripts.
4. **Keyword Driven Testing Framework:** The Keyword Driven testing framework is an extension to Data-driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.
5. **Hybrid Testing Framework:** Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.
6. **Behavior Driven Development Framework:** Behavior Driven Development framework allows automation of functional validations in an easily readable and understandable format to Business Analysts, Developers, Testers, etc
7. **Datadriven framework:**  In this framework, the test data is separated and kept outside the Test Scripts, while[Test Case](https://www.guru99.com/test-case.html)logic resides in Test Scripts.  Test data is read from the external files ( Excel Files) and are loaded into the variables inside the Test Script.  Variables are used for both for input values and for verification values.
8. **Keyworddriven framework:** The keyword driven frameworks requires the development of data tables and keywords, independent of the test automation.  In a keyword driven test, the functionality of the application under test is documented in a table as well as step by step instructions for each test.

**Q #50) What is Object Repository? How can we create an Object Repository in Selenium?**

Object Repository is a term used to refer to the collection of web elements belonging to Application Under Test (AUT) along with their locator values. Thus, whenever the element is required within the script, the locator value can be populated from the Object Repository. Object Repository is used to store locators in a centralized location instead of hardcoding them within the scripts.

You can use…..driver.findElements(By.xpath(“//iframe”))….

This will return list of frames.

**How to check isElementPresent**

protected boolean isElementPresent(By by){

try{

driver.findElement(by);

return true;

}

catch(NoSuchElementException e){

return false;

}

}

1. webDriver.findElement(By.xpath("//\*[@id='element']")).isDisplayed();
2. driver.findElement(By.cssSelector("a > font")).isEnabled();
3. driver.getPageSource().contains("Text to check");

#### 90) Mention what are the challenges in Handling Ajax Call in Selenium Webdriver?

The challenges faced in Handling Ajax Call in Selenium Webdriver are

* Using "pause" command for handling Ajax call is not completely reliable. Long pause time makes the test unacceptably slow and increases the testing time. Instead, "waitforcondition" will be more helpful in testing Ajax applications.
* It is difficult to assess the risk associated with particular Ajax applications
* Given full freedom to developers to modify Ajax application makes the testing process challenging
* Creating automated test request may be difficult for testing tools as such AJAX application often use different encoding or serialization technique to submit POST data.

#### 92) Mention in what ways you can customize TestNG report?

You can customize TestNG report in two ways,

* Using ITestListener Interface
* Using IReporter Interface

#### 94) Mention what is Listeners in Selenium WebDriver?

In Selenium WebDriver, Listeners "listen" to the event defined in the selenium script and behave accordingly. It allows customizing TestNG reports or logs. There are two main listeners i.e. WebDriver Listeners and TestNG Listeners.

#### 95) Mention what are the types of [Listeners in TestNG](https://www.guru99.com/listeners-selenium-webdriver.html)?

The types of Listeners in TestNG are,

* IAnnotationTransformer
* IAnnotationTransformer2
* IConfigurable
* IConfigurationListener
* IExecutionListener
* IHookable
* IInvokedMethodListener
* IInvokedMethodListener2
* IMethodInterceptor
* IReporter
* ISuiteListener
* ITestListener

**Alternative click method : using Javascript**

WebElement element = driver.findElement(By.id("gbqfd"));

JavascriptExecutor executor = (JavascriptExecutor)driver;

executor.executeScript("arguments[0].click();", element);

**Alternative click method: using Action class**

Actions builder = new Actions(driver);

builder.moveToElement( el ).click( el );

builder.perform();

**Common Exceptions in Selenium Web driver**

TimeoutException Thrown when there is not enough time for a command to be completed. For Example, the element searched wasn't found in the specified time.

WebDriverException This Exception takes place when the WebDriver is acting right after you close the browser.

ConnectionClosedException This type of Exception takes place when there is a disconnection in the driver.

NoSuchElementException This Exception occurs if an element could not be found.

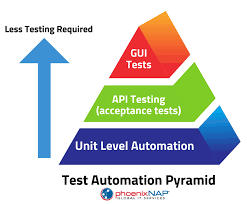
NoAlertPresentException This Exception occurs when you switch to no presented alert.

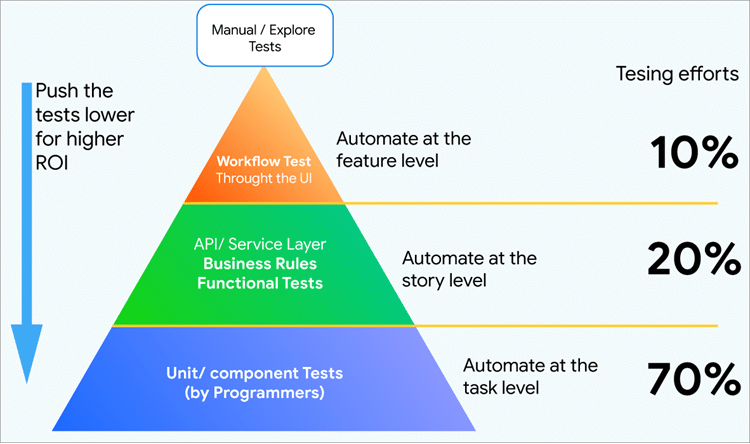
NoSuchWindowException This Exception occurs if the window target to be switch does not exist.

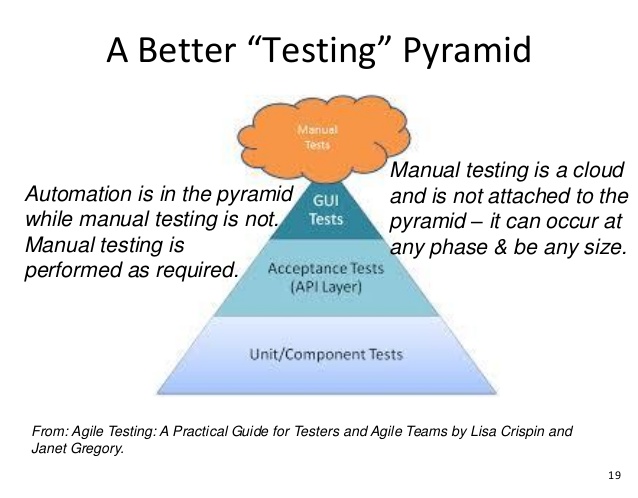
ElementNotVisibleException This type of Selenium exception occurs when an existing element in DOM has a feature set as hidden.

NoSuchAttributeException This kind of Exception occurs when the attribute of an element could not be found.

StaleElementReferenceException This Selenium exception occurs happens when the web element is detached from the current DOM.







**Framework features:**

**Implemented Retry Analyzer** – This will set how many time s to execute the failed test cases

Implemented Hybrid Framework which combines the **Module Based Testing Framework and Library Architecture Testing Framework**

1. **Module Based Testing Framework:** The framework divides the entire “Application Under Test” into the number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts have taken together builds a larger test script representing more than one module.
2. **Library Architecture Testing Framework:** The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.

**Framework supported the database connectivity / capable of doing database testing**

**Implemented Logger mechanisum** using TestListenerAdapter and Reporter class / **customized Test NG reporting**

**Implemented machanisum to take screenshot in case of test failure**

**Implemented ZAP integration**

**Integrated with CI/CD Pipeline – Jenkins**

**Implemented Headless Execution- added plugin xvfb configuration**

**Integrated Code review tools(Static analyzer, PMD, CPD tools) – PMD(Programming Mistake Detector) with plugin added CPD(Copy Paste Detector)**

**Had flexibility of changing the locator, but not changing the code** (Keeping object- repository outside of jar file)

Common Issues in Slenium Automation

1. org.openqa.selenium.firefox.NotConnectedException: Unable to connect to host 127.0.0.1 on port 7055 after 45000 ms

Solution: webdriver issue, need to upgrade it to latest version

1. StaleElement Exception occurred

Solution: Handle the element with wait methods (Explicit wait)

1. Fleakiness- Timeout issues

Solution: Use Explicit wait

1. Testing Multi Tab

Solution:

String newWindow = driver.getWindowHandle();

driver.findElement(By.cssSelector(“body”)).sendKeys(Keys.CONTROL+”t”);

driver.get("Second URL");

driver.switchTo().window(mainWindow);

1. Limited Reporting

Solution: Achived Through TestNg, Reporter class, ItestListener to customize the report

1. Handling Popups

Solution: Handling Windows based popup have another problem, achinved through Third party tool(AutoIT).

1. We can’t use selenium to test local windows based applications.

Solution: Use some other tools AutoIT, WinApp Driver, Winium, ZAPTEST.

1. Not possible to handle captcha and Barcode reader using selenium
2. Handling Dynamic web element:

Solution : Using xpath dynamically, means on the go change the xpath and use it.

## What is Page Object Model?

**Page Object Model (POM)** is a design pattern, popularly used in test automation that creates **Object Repository** for web UI elements. The advantage of the model is that it reduces code duplication and improves test maintenance.

Under this model, for each web page in the application, there should be a corresponding **Page Class.**This Page class will identify the WebElements of that web page and also contains **Page methods** which perform operations on those WebElements. Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay().

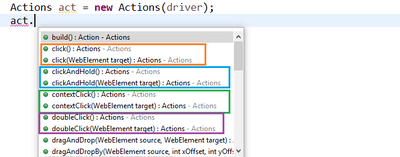
**Where we used Compile time polymorphism in Selenium:**

//driver.switchTo().frame(String id or Name), frame(Webelement element)

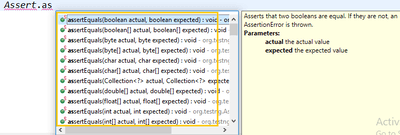
The first frame(String id or Name) accepts a String parameter, but the second frame(WebElement element) accepts the Webelement as a parameter.

When there is more than one method with the same name, and those methods accept either different types of parameters or different number parameter, then it is called as Method Overloading.

2. We use overriding in Action class methods also, some will take no parameter, some will take parameter



1. Almost all the methods present in the [Assert class in TestNG](https://chercher.tech/java/testng-assertions-selenium-webdriver).



**Method Overriding in Selenium/ Framework:**

**public** **class** CustomListener **extends** TestListenerAdapter

**here**

@Override

**public** **void** onTestFailure(ITestResult tr) {

log(tr.getName()+ "--Test method failed\n");

}

@Override

**public** **void** onTestSkipped(ITestResult tr) {

log(tr.getName()+ "--Test method skipped\n");

}

@Override

**public** **void** onTestSuccess(ITestResult tr) {

log(tr.getName()+ "--Test method success\n");

}

These methods imeplentation is overridden

