# 100 Most Advanced Selenium Interview Questions And Answers

## 1. What is Automation Testing?

Automation testing is the process of testing a software or application using an automation testing tool to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools. It is required when we have huge amount of regression test cases. Some most popular tools to do automation testing are HP QTP/UFT, Selenium WebDriver, etc.,

## 2. What are the benefits of Automation Testing?

This is one of the common interview questions in any Automation testing job.

1. Saves time and money. Automation testing is faster in execution.
2. Reusability of code. Create one time and execute multiple times with less or no maintenance.
3. Easy reporting. It generates automatic reports after test execution.
4. Easy for compatibility testing. It enables parallel execution in the combination of different OS and browser environments.
5. Low-cost maintenance. It is cheaper compared to manual testing in a long run.
6. Automated testing is more reliable.
7. Automated testing is more powerful and versatile. Automation tools allow us to integrate with Cross Browser Testing Tools, Jenkins, Github etc.,
8. It is mostly used for regression testing. Supports execution of repeated test cases.
9. Minimal manual intervention. Test scripts can be run unattended.
10. Maximum coverage. It helps to increase the test coverage.

## 3. What are the challenges and limitations of Selenium WebDriver?

As we all know Selenium WebDriver is a tool which automates the browser to mimic real user actions on the web. Selenium is a free open source testing tool. Some of the challenges with selenium webdriver are as follows

1. We cannot test windows application
2. We cannot test mobile apps
3. Limited reporting

4. Handling dynamic Elements
5. Handling page load
6. Handling pop up windows
7. Handling captcha

## 4. What type of tests have you automated?

Our main focus is to automate test cases to do *Regression testing, Smoke testing, and Sanity testing.* Sometimes based on the project and the test time estimation, we do focus on End to End testing.

## 5. How many test cases you have automated per day?

It depends on Test case scenario complexity and length. I did automate 2-5 test scenarios per day when the complexity is limited. Sometimes just 1 or fewer test scenarios in a day when the complexity is high.

## 6. What is a Framework*?*

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results. There are different types of automation frameworks and the most common ones are:

- Data Driven Testing Framework
- Keyword Driven Testing Framework
- Hybrid Testing Framework

## 7. Have you created any Framework?

*If you are a beginner:* You can say "No, I didn't get a chance to create framework from the scratch. I have used the framework which is already available. My contribution is mostly in creating test cases by using the existing framework."

*If you are a beginner but have good knowledge on creating framework:* You can say "Yes, I have involved in developing framework along with other automation tester in my company."

*If you are an experienced tester:* You can say "I have contributed in developing framework." or You can say "Yes, I have created framework from

the scratch. There was no automation process in my previous company. I designed the framework from the scratch."

Why WebDriver driver = new FirefoxDriver();

## 8. Can you explain the Framework which you have used in your Selenium Project?

Here we have clearly explained each component of Framework.

## 9. Why do you prefer Selenium Automation Tool?

1. Free and open source
2. Have large user base and helping communities
3. Cross-browser compatibility
4. Platform compatibility
5. Multiple programming languages support such as Java, Perl, Phyton, Php, C# etc.,

## 10. What is Selenium?

Selenium is an open source (free) automated testing suite to test web applications. It supports different platforms and browsers. It has gained a lot of popularity in terms of web-based automated testing and giving a great competition to the famous commercial tool HP QTP (Quick Test Professional) AKA HP UFT (Unified Functional Testing).

Selenium is a set of different software tools. Each tool has a different approach in supporting web based automation testing.

It has four components namely,

1. Selenium IDE (Selenium Integrated Development Environment)
2. Selenium RC (Selenium Remote Control)
3. Selenium WebDriver
4. Selenium Grid

## 11. What is Selenium IDE?

Selenium IDE (Integrated Development Environment) is a Firefox plugin. It is the simplest framework in the Selenium Suite. It allows us to record and

playback the scripts. Even though we can create scripts using Selenium IDE, we need to use Selenium RC or Selenium WebDriver to write more advanced and robust test cases.

## 12. What is Selenese?

Selenese is the language which is used to write test scripts in Selenium IDE.

## 13. Which is the only browser that supports Selenium IDE to be used?

Firefox

## 14. What is Selenium RC?

Selenium RC AKA Selenium Remote control / Selenium 1. Selenium Remote Control was the main Selenium project for a long time before the WebDriver merge brought up Selenium 2. Selenium 1 is still actively supported (in maintenance mode). It relies on JavaScript for automation. It supports Java, Javascript, Ruby, PHP, Python, Perl and C#. It supports almost every browser out there.

## 15. What is Selenium WebDriver?

Selenium WebDriver AKA Selenium 2 is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It controls the browser by directly communicating with it. Selenium WebDriver supports Java, C#, PHP, Python, Perl, Ruby.

## 16. What is Selenium Grid?

Selenium Grid is a tool used together with Selenium RC to run tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems.

In simple words, it is used to distribute your test execution on multiple platforms and environments concurrently.

## 17. When do you use Selenium Grid?

Selenium Grid can be used to execute same or different test scripts on multiple platforms and browsers concurrently so as to achieve distributed test execution

## 18. What are the advantages of Selenium Grid?

It allows running test cases in parallel thereby saving test execution time.
It allows multi-browser testing
It allows us to execute test cases on multi-platform

## 19. What is a hub in Selenium Grid?

A hub is a server or a central point that controls the test executions on different machines.

## 20. What is a node in Selenium Grid?

Node is the machine which is attached to the hub. There can be multiple nodes in Selenium Grid.

## 21. What are the types of WebDriver APIs available in Selenium?

- Firefox Driver
- Gecko Driver
- InternetExplorer Driver
- Chrome Driver
- HTMLUnit Driver
- Opera Driver
- Safari Driver
- Android Driver
- iPhone Driver
- EventFiringWebDriver

## 22. Which WebDriver implementation claims to be the fastest?

The fastest implementation of WebDriver is the HTMLUnitDriver. It is because the HTMLUnitDriver does not execute tests in the browser.

How To Do Headless Browser Testing using Selenium WebDriver

## 23. What are the Programming Languages supported by Selenium WebDiver?

- Java
- C#
- Python
- Ruby
- Perl
- PHP

## 24. What are the Operating Systems supported by Selenium WebDriver?

- Windows
- Linux
- Apple

## 25. What are the Open-source Frameworks supported by Selenium WebDriver?

- JUnit
- TestNG

## 26. What are the Locators available in Selenium?

In Selenium WebDriver, there are 8 different types of locators:

1. ID –
2. ClassName –
3. Name –
4. TagName –
5. LinkText –
6. PartialLinkText –
7. XPath
8. CSS Selector –

## 27. What is an XPath?

XPath is used to locate the elements. Using XPath, we could navigate through elements and attributes in an XML document to locate web elements such as textbox, button, checkbox, Image etc., in a web page.

Learn How To Write Dynamic XPath

## 28. What is the difference between "/" and "//"

**Single Slash "/" –** Single slash is used to create XPath with absolute path i.e. the XPath would be created to start selection from the document node/start node.

**Double Slash "//" –** Double slash is used to create XPath with relative path i.e. the XPath would be created to start selection from anywhere within the document.

## 29. What is the difference between Absolute Path and Relative Path?

Absolute XPath starts from the root node and ends with desired descendant element's node. It starts with top HTML node and ends with input node. It starts with a single forward slash(/) as shown below

```
1 /html/body/div[3]/div[1]/form/table/tbody/tr[1]/td/input
```

Relative XPath starts from any node in between the HTML page to the current element's node(last node of the element). It starts with a double forward slash(//) as shown below.

```
1 //input[@id='email']
```

## 30. What is the difference between Assert and Verify in Selenium?

**Assert:** In simple words, if the assert condition is true then the program control will execute the next test step but if the condition is false, the execution will stop and further test step will not be executed.

**Verify:** In simple words, there won't be any halt in the test execution even though the verify condition is true or false.

## 31. What are Soft Assert and Hard Assert in Selenium?

Soft Assert: Soft Assert collects errors during `@Test` Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

Hard Assert: Hard Assert throws an *AssertException* immediately when an assert statement fails and test suite continues with next `@Test`

## 32. What are the verification points available in Selenium?

In Selenium IDE, we use Selenese Verify and Assert Commands as Verification points
In Selenium WebDriver, there is no built-in features for verification points. It totally depends on our coding style. some of the Verification points are

- To check for page title
- To check for certain text
- To check for certain element (text box, button, drop down, etc.)

## 33. How to launch a browser using Selenium WebDriver?

WebDriver is an Interface. We create an Object of a required driver class such as FirefoxDriver, ChromeDriver, InternetExplorerDriver etc.,

*To launch Firefox Driver:*
WebDriver driver = new FirefoxDriver();

Note: *If you use geckodriver with Selenium, you must upgrade to Selenium 3.3. Here we have to set the property as follows*

```
System.setProperty("webdriver.gecko.driver", "D:\\Selenium
Environment\\Drivers\\geckodriver.exe");
```
1

*To launch Chrome Driver:*
WebDriver driver = new ChromeDriver();

*To launch Internet Explorer Driver:*
WebDriver driver = new InternetExplorerDriver();

*To launch Safari Driver:*
WebDriver driver = new SafariDriver();

## 34. Is the FirefoxDriver a Class or an Interface?

*FirefoxDriver* is a Java class, and it implements the *WebDriver* interface.

## 35. What is the super interface of WebDriver?

SearchContext.

## 36. Explain the line of code *Webdriver driver = new FirefoxDriver(); ?*

```
1 Webdriver driver = new FirefoxDriver();
```

'*WebDriver*' is an interface and we are creating an object of type WebDriver instantiating an object of FirefoxDriver class.

## 37. We do create a reference variable 'driver' of type WebDriver

```
1 WebDriver driver = new FirefoxDriver();

2

3 instead of creating

4

5 FirefoxDriver driver = new FirefoxDriver();
```

## What is the purpose of doing this way?

If we create a reference variable driver of type WebDriver then we could use the same driver variable to work with any browser of our choice such as IEDriver, SafariDriver etc.,

## 38. What are the different exceptions you have faced in Selenium WebDriver?

Some of the exceptions I have faced in my current project are

1. ElementNotVisibleException
2. StaleElementReferenceException

**Element Not visible Exception:**

This exception will be thrown when you are trying to locate a particular element on webpage that is not currently visible eventhough it is present in the DOM. Also sometimes, if you are trying to locate an element with the xpath which associates with two or more element.

**Stale Element Reference Exception:**

A stale element reference exception is thrown in one of two cases, the first being more common than the second.

The two reasons for Stale element reference are

1. The element has been deleted entirely.
2. The element is no longer attached to the DOM.

We face this stale element reference exception when the element we are interacting is destroyed and then recreated again. When this happens the reference of the element in the DOM becomes stale. Hence we are not able to get the reference to the element.

Some other exceptions we usually face are as follows:

- WebDriverException
- IllegalStateException
- TimeoutException
- NoAlertPresentException
- NoSuchWindowException
- NoSuchElementException

**39. How to Login into any site if it is showing an Authentication Pop-Up for Username and Password?**

1 http://username:password@url

2 e.g. http://myUserName:myPassword@softwaretestingmaterial.com

**40. What are the types of waits available in Selenium WebDriver?**

In Selenium we could see three types of waits such as Implicit Waits, Explicit Waits and Fluent Waits.

- Implicit Waits –
- Explicit Waits –
-  Fluent Waits

## 41. What is Implicit Wait In Selenium WebDriver?

Implicit waits tell to the WebDriver to wait for a certain amount of time before it throws an exception. Once we set the time, WebDriver will wait for the element based on the time we set before it throws an exception. The default setting is 0 (zero). We need to set some wait time to make WebDriver to wait for the required time.

Practical example

## 42. What is WebDriver Wait In Selenium WebDriver?

WebDriverWait is applied on a certain element with defined *expected condition*and *time*. This wait is only applied to the specified element. This wait can also throw an exception when an element is not found.

## 43. What is Fluent Wait In Selenium WebDriver?

FluentWait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an "*ElementNotVisibleException*" exception.

## 44. How to input text in the text box using Selenium WebDriver?

By using sendKeys() method

```
1 WebDriver driver = new FirefoxDriver();

2 driver.get("https://www.gmail.com");

3 driver.findElement(By.xpath("xpath")).sendKeys("Software Testing Material Website");
```

## 45. How to input text in the text box without calling the sendKeys()?

```
1 // To initialize js object

2 JavascriptExecutor JS = (JavascriptExecutor)webdriver;

3 // To enter username

4 JS.executeScript("document.getElementById('User').value='SoftwareTestingMaterial.com'");

5 // To enter password

6 JS.executeScript("document.getElementById('Pass').value='tester'");
```

Learn JavaScriptExecutor in Selenium with Examples

## 46. How to clear the text in the text box using Selenium WebDriver?

By using clear() method

```
1   WebDriver driver = new FirefoxDriver();

2   driver.get("https://www.gmail.com");

3   driver.findElement(By.xpath("xpath_of_element1")).sendKeys("Software Testing Material

4   Website");

    driver.findElement(By.xpath("xpath_of_element1")).clear();
```

## 47. How to get a text of a web element?

By using getText() method

```
1  package softwareTestingMaterial;

2  import org.openqa.selenium.By;

3  import org.openqa.selenium.WebDriver;

4  import org.openqa.selenium.chrome.ChromeDriver;

5  import org.testng.annotations.Test;

6  public class TestTestTest {

7
```

```
 8   @Test

 9   public void testmethod(){

10  System.setProperty("webdriver.chrome.driver", "D:\\Selenium

11 Environment\\Drivers\\chromedriver.exe");

12     WebDriver driver = new ChromeDriver();

13     driver.get("https://www.google.com");

14     String availableText =

15 driver.findElement(By.xpath("//*[@id='gbw']/div/div/div[1]/div[1]/a")).getText();

16     System.out.println("Text Available is :"+availableText);

    }

    }
```

## 48. How to get an attribute value using Selenium WebDriver?

By using getAttribute(value);

It returns the value of the attribute passed as a parameter.

HTML:

```
1 <input name="nameSelenium" value="valueSelenium">SoftwareTestingMaterial</input>
```

Selenium Code:

```
1 String attributeValue = driver.findElement(By.name("nameSelenium")).getAttribute("value");

2 System.out.println("Available attribute value is :"+attributeValue);

3 Output: valueSelenium
```

## 49. How to click on a hyperlink using Selenium WebDriver?

We use click() method in Selenium to click on the hyperlink

```
1 driver.findElement(By.linkText("Software Testing Material Website")).click();
```

## 50. How to submit a form using Selenium WebDriver?

We use "submit" method on element to submit a form

```
1 driver.findElement(By.id("form_1")).submit();
```

Alternatively, you can use click method on the element which does form submission

## 51. How to press ENTER key on text box In Selenium WebDriver?

To press ENTER key using Selenium WebDriver, We need to use Selenium Enum Keys with its constant ENTER.

```
1 driver.findElement(By.xpath("xpath")).sendKeys(Keys.ENTER);
```

## 52. How to pause a test execution for 5 seconds at a specific point?

By using **java.lang.Thread.sleep***(long milliseconds)* method we could pause the execution for a specific time. To pause 5 seconds, we need to pass parameter as 5000 (5 seconds)

```
1 Thread.sleep(5000)
```

## 53. Is Selenium Server needed to run Selenium WebDriver Scripts?

When we are distributing our Selenium WebDriver scripts to execute using Selenium Grid, we need to use Selenium Server.

## 54. What happens if I run this command. *driver.get("www.softwaretestingmaterial.com") ;*

An exception is thrown. We need to pass HTTP protocol within driver.get() method.

```
1 driver.get("https://www.softwaretestingmaterial.com");
```

## 55. What is the alternative to *driver.get()* method to open an URL using Selenium WebDriver?

Alternative method to *driver.get("url")* method is *driver.navigate.to("url")*

## 56. What is the difference between *driver.get()* and *driver.navigate.to("url")*?

*driver.get():* To open an URL and it will wait till the whole page gets loaded
*driver.navigate.to():* To navigate to an URL and It will not wait till the whole page gets loaded

## 57. Can I navigate back and forth in a browser in Selenium WebDriver?

We use Navigate interface to do navigate back and forth in a browser. It has methods to move back, forward as well as to refresh a page.

**driver.navigate().forward();** – to navigate to the next web page with reference to the browser's history
**driver.navigate().back();** – takes back to the previous webpage with reference to the browser's history
**driver.navigate().refresh();** – to refresh the current web page thereby reloading all the web elements
**driver.navigate().to("url");** – to launch a new web browser window and navigate to the specified URL

## 58. What are the different types of navigation commands?

Refer above question (Can I navigate back and forth in a browser)

## 59. How to fetch the current page URL in Selenium?

To fetch the current page URL, we use *getCurrentURL()*

```
1 driver.getCurrentUrl();
```

## 60. How can we maximize browser window in Selenium?

To maximize browser window in selenium we use *maximize()* method. This method maximizes the current window if it is not already maximized

```
1 driver.manage().window().maximize();
```

## 61. How to delete cookies in Selenium?

To delete cookies we use *deleteAllCookies()* method

 1 driver.manage().deleteAllCookies();

## 62. What are the ways to refresh a browser using Selenium WebDriver?

There are multiple ways to refresh a page in selenium

- Using *driver.navigate().refresh()* command as mentioned in the question 45
- Using driver.get("URL") on the current URL or using *driver.getCurrentUrl()*
- Using driver.navigate().to("URL") on the current URL or *driver.navigate().to(driver.getCurrentUrl());*
- Using *sendKeys(Keys.F5)* on any textbox on the webpage

## 63. What is the difference between driver.getWindowHandle() and driver.getWindowHandles() in Selenium WebDriver?

*driver.getWindowHandle()* – It returns a handle of the current page (a unique identifier)
*driver.getWindowHandles()* – It returns a set of handles of the all the pages available.

## 64. What is the difference between driver.close() and driver.quit() methods?

Purpose of these two methods (driver.close and driver.quit) is almost same. Both allow us to close a browser but still, there is a difference.

*driver.close():* To close current WebDriver instance
*driver.quit():* To close all the opened WebDriver instances

## 65. What is the difference between driver.findElement() and driver.findElements() commands?

The difference between driver.findElement() and driver.findElements() commands is-

- findElement() returns a single WebElement (found first) based on the locator passed as parameter. Whereas findElements() returns a list of WebElements, all satisfying the locator value passed.
- Syntax of findElement()-
  WebElement textbox = driver.findElement(By.id("textBoxLocator"));
  Syntax of findElements()-
  List <WebElement> elements = element.findElements(By.id("value"));
- Another difference between the two is- if no element is found then findElement() throws NoSuchElementException whereas findElements() returns a list of 0 elements.

## 66. How to find whether an element is displayed on the web page?

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.

isDisplayed()

```
1 boolean elePresent = driver.findElement(By.xpath("xpath")).isDisplayed();
```

isSelected()

```
1 boolean eleSelected= driver.findElement(By.xpath("xpath")).isSelected();
```

isEnabled()

```
1 boolean eleEnabled= driver.findElement(By.xpath("xpath")).isEnabled();
```

## 67. How to select a value in a dropdown?

By using *Select* class

```
1 WebElement mySelectElement = driver.findElement(By.name("dropdown"));
```

```
2 Select dropdown = new Select(mySelectElement);
```

```
3 dropdown.selectByVisibleText(Text);
```

```
4 dropdown.selectByIndex(Index);
```

5 dropdown.selectByValue(Value);

Practical Example:

## 68. How to capture Screenshot in Selenium WebDriver?

Test cases may fail while executing the test scripts. While we are executing the test cases manually we just take a screenshot and place in a result repository. The same can be done by using Selenium WebDriver.

Some of the scenarios we may need to capture a screenshot using Selenium WebDriver are

i. Application issues
ii. Assertion Failure
iii. Difficulty to find Webelements on the web page
iv. Timeout to find Webelements on the web page

Selenium provides an interface called *TakesScreenshot* which has a method *getScreenShotAs* which can be used to take a screenshot of the application under test.

In Selenium 3, we may face few issues while capturing Screenshots. To overcome we use aShot utility. Click on below links to see posts related to the normal way of capturing a screenshot and capturing a screenshot using aShot utility.

Capture screenshot using Selenium WebDriver

Full Page Screenshot using aShot utility

Failed Test Cases Screenshot

## 69. How to mouse hover on a web element using WebDriver?

By using Actions class

1 WebElement ele = driver.findElement(By.xpath("xpath"));

2 //Create object 'action' of an Actions class

3 Actions action = new Actions(driver);

4 //Mouseover on an element

5 action.moveToElement(ele).perform();

Practical Example.

## 70. How can we handle web based pop-up?

To handle alerts popups we need to do switch to the alert window and call Selenium WebDriver Alert API methods.

Practical Example.

## 71. How can we handle windows based pop up?

Selenium doesn't support windows based applications. It is an automation testing tool which supports only web application testing. We could handle windows based popups in Selenium using some third party tools such as AutoIT, Robot class etc.

## 72. How to handle hidden elements in Selenium WebDriver?

It is one of the most important selenium interview questions.

We can handle hidden elements by using javaScript executor

```
1   (JavascriptExecutor(driver)).executeScript("document.getElementsByClassName(ElementLoc
    ator).click();");
```

## 73. How can you find Broken Links in a page using Selenium WebDriver?

## 74. How to find more than one web element in the list?

1  // To store the list

```
2  List <WebElement> eleList = driver.findElements(By.xpath("xpath"));

3  // To fetch the size of the list

4  int listSize = eleList.size();

5  //for loop

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

7  {

8   // Clicking on each link

9   links.get(i).click();

10  // Navigating back to the previous page that stores the links

11  driver.navigate().back();

12 }
```

## 75. How to read a JavaScript variable in Selenium WebDriver?

By using JavascriptExecutor

```
1 // To initialize the JS object.

2 JavascriptExecutor JS = (JavascriptExecutor) webdriver;

3 // To get the site title.

4 String title = (String)JS.executeScript("return document.title");

5 System.out.println("Title of the webpage : " + title);
```

## 76. What is JavaScriptExecutor and in which cases JavaScriptExecutor will help in Selenium automation?

In general, we click on an element using click() method in Selenium.

```
1 driver.findElement(By.id("Id Value")).click();
```

Sometimes web controls don't react well against selenium commands and we may face issues with the above statement (click()). To overcome such kind of situation, we use JavaScriptExecutor interface.

It provides a mechanism to execute Javascript through Selenium driver. It provides "executescript" & "executeAsyncScript" methods, to run JavaScript in the context of the currently selected frame or window.

There is no need to write a separate script to execute JavaScript within the browser using Selenium WebDriver script. Just we use predefined interface named 'Java Script Executor'. We need to import the JavascriptExecutor package in the script.

```
1 JavascriptExecutor js = (JavascriptExecutor) driver;

2 js.executeScript(Script,Arguments);
```

*Script* – The JavaScript to execute
*Arguments* – The arguments to the script(Optional). May be empty.
*Returns* – One of Boolean, Long, String, List, WebElement, or null.

Let's see some scenarios we could handle using this Interface:

1. To type Text in Selenium WebDriver without using sendKeys() method
2. To click a Button in Selenium WebDriver using JavaScript
3. To handle Checkbox
4. To generate Alert Pop window in selenium
5. To refresh browser window using Javascript
6. To get innertext of the entire webpage in Selenium
7. To get the Title of our webpage
8. To get the domain
9. To get the URL of a webpage
10. To perform Scroll on an application using  Selenium
11. To click on a SubMenu which is only visible on mouse hover on Menu
12. To navigate to different page using Javascript

**77. How do you read test data from excels?**

Test data can efficiently be read from excel using JXL or POI API. POI API has many advantages than JXL.
Click here to see a practical example of using Apache POI.

## 78. Is it possible to automate the captcha using Selenium?

No, It's not possible to automate captcha and bar code reader.

## 79. How to handle Ajax calls in Selenium WebDriver?

Handling AJAX calls is one of the common issues when using Selenium WebDriver. We wouldn't know when the AJAX call would get completed and the page has been updated. In this post, we see how to handle AJAX calls using Selenium.

AJAX stands for Asynchronous JavaScript and XML. AJAX allows the web page to retrieve small amounts of data from the server without reloading the entire page. AJAX sends HTTP requests from the client to server and then process the server's response without reloading the entire page. To handle AJAX controls, wait commands may not work. It's just because the actual page is not going to refresh.

When you click on a submit button, the required information may appear on the web page without refreshing the browser. Sometimes it may load in a second and sometimes it may take longer. We have no control over loading time. The best approach to handle this kind of situations in selenium is to use dynamic waits (i.e. WebDriverWait in combination with ExpectedCondition)

Some of the methods which are available are as follows:

1. titleIs() – The expected condition waits for a page with a specific title.

```
1 wait.until(ExpectedConditions.titleIs("Deal of the Day"));
```

2. elementToBeClickable() – The expected condition waits for an element to be clickable i.e. it should be present/displayed/visible on the screen as well as enabled.

```
1 wait.until(ExpectedConditions.elementToBeClickable(By.xpath("xpath")));
```

3. alertIsPresent() – The expected condition waits for an alert box to appear.

```
1 wait.until(ExpectedConditions.alertIsPresent()) !=null);
```

4. textToBePresentInElement() – The expected condition waits for an element having a certain string pattern.

wait.until(ExpectedConditions.textToBePresentInElement(By.id("title"), "text to be
1
found"));

## 80. List some scenarios which we cannot automate using Selenium WebDriver?

1. Bitmap comparison is not possible using Selenium WebDriver
2. Automating Captcha is not possible using Selenium WebDriver
3. We can not read bar code using Selenium WebDriver

## 81. What is Object Repository in Selenium WebDriver?

Object Repository is used to store element locator values in a centralized location instead of hard coding them within the scripts. We do create a property file (*.properties*) to store all the element locators and these property files act as an object repository in Selenium WebDriver.

## 82. How you build Object Repository in your project?

In QTP, there is an Object Repository concept. When a user records a test, the objects and its properties are captured by default in an Object Repository. QTP uses this Object Repository to play back the scripts. Coming to Selenium, there is no default Object Repository concept. It doesn't mean that there is no Object Repository in Selenium. Even though there is no default one still we could create our own. In Selenium, we call objects as locators (such as ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, XPath, and CSS). Object repository is a collection of objects. One of the ways to create Object Repository is to place all the locators in a separate file (i.e., properties file). But the best way is to use Page Object Model. In the Page Object Model Design Pattern, each web page is represented as a class. All the objects related to a particular page of a web application are stored in a class.

## 83. What is Page Object Model in Selenium?

Page Object Model is a Design Pattern which has become popular in Selenium Test Automation. It is widely used design pattern in Selenium for enhancing test maintenance and reducing code duplication. Page object

model (POM) can be used in any kind of framework such as modular, data-driven, keyword driven, hybrid framework etc. A page object is an object-oriented class that serves as an interface to a page of your Application Under Test(AUT). The tests then use the methods of this page object class whenever they need to interact with the User Interface (UI) of that page. The benefit is that if the UI changes for the page, the tests themselves don't need to change, only the code within the page object needs to change. Subsequently, all changes to support that new UI is located in one place.

## 84. What is Page Factory?

We have seen that 'Page Object Model' is a way of representing an application in a test framework. For every 'page' in the application, we create a Page Object to reference the 'page' whereas a 'Page Factory' is one way of implementing the 'Page Object Model'.

## 85. What is the difference between Page Object Model (POM) and Page Factory?

Page Object is a class that represents a web page and hold the functionality and members.
Page Factory is a way to initialize the web elements you want to interact with within the page object when you create an instance of it.

## 86. What are the advantages of Page Object Model Framework?

**Code reusability** – We could achieve code reusability by writing the code once and use it in different tests.

**Code maintainability** – There is a clean separation between test code and page specific code such as locators and layout which becomes very easy to maintain code. Code changes only on Page Object Classes when a UI change occurs. It enhances test maintenance and reduces code duplication.

**Object Repository** – Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.

**Readability** – Improves readability due to clean separation between test code and page specific code

## 87. How can you use the Recovery Scenario in Selenium WebDriver?

By using "*Try Catch Block*" within Selenium WebDriver Java tests.

1 try {

2     driver.get("www.SoftwareTestingMaterial.com");

3 }catch(Exception e){

4     System.out.println(e.getMessage());

5 }

## 88. How to Upload a file in Selenium WebDriver?

There are two cases which are majorly used to upload a file in Selenium WebDriver such as using *SendKeys* Method and using *AutoIT* Script.

Practical Example.

## 89. How to Download a file in Selenium WebDriver?

By using *AutoIT* script, we could download a file in Selenium WebDriver.

Practical Example

## 90. How to run Selenium WebDriver Test from the command line?

Run Java Program using Command Prompt

Run TestNG using Command Prompt

## 91. How to switch between frames in Selenium?

By using the following code, we could switch between frames.

1 driver.switchTo().frame();

## 92. How to connect a Database in selenium?

As we all know Selenium WebDriver is a tool to automate User Interface. We could only interact with Browser using Selenium WebDriver.

We use JDBC Driver to connect the Database in Selenium (While using Java Programming Language).

Practical Example

## 93. How To Resize Browser Window Using Selenium WebDriver?

To resize the browser window to particular dimensions, we use 'Dimension' class to resize the browser window.

Practical Example

## 94. How To Scroll Web Page Down Or UP Using Selenium WebDriver?

JavaScript **scrollBy**() method scrolls the document by the specified number of pixels.

Practical Example

## 95. How To Perform Right Click Action (Context Click) In Selenium WebDriver?

We use Actions class in Selenium WebDriver to do Right-Click (Context Click) action.

Practical Example

## 96. How To Perform Double Click Action In Selenium WebDriver?

We use Actions class to do Double click action in selenium.

Practical Example

## 97. How To Perform Drag And Drop Action in Selenium WebDriver?

In some applications, we may face a situation to automate drag and drop an item from one location to another location. We could not achieve these using basic elements. Selenium has provided an "Actions" class to handle this kind of scenarios. We overcome this kind of scenarios such as drag and drop using Actions Class.

To achieve this, we use Actions class in Selenium WebDriver.

Practical Example

## 98. How To Highlight Element Using Selenium WebDriver?

By using JavascriptExecutor interface, we could highlight the specified element

Practical Example

## 99. Have you used any crossbrowsertesting tool to run selenium scripts on cloud?

I have used BrowserStack to run selenium tests on multiple browsers & Multiple operating systems in parallel. Earlier we have made a video on how to use BrowserStack to run selenium scripts on the cloud. Find the link in the description below.

## 100. What is desired capabilities?

In Selenium we use desired capabilities to handle SSL certificates in chrome browser

We need to create an instance of DesiredCapabilities

```
1 DesiredCapabilities desiredCapability = DesiredCapabilities.chrome();
```

## 101. What is Continuous Integration?

Continuous Integration is abbreviated as CI. Continuous Integration is a development practice which aims to make sure the correctness of a software. After each commit, a suite of tests run automatically and test the

software to ensure whether the software is running without any breaks. If any test fails, we will get immediate feedback say "build is broken".

In simple words, continuous integration is a process of verifying the correctness of a software.

Some of the continuous integration tools are Jenkins, TeamCity, Bamboo, Travis, Circle Ci, Bitbucket.

We can schedule the test suite execution using these CI Tools.

## 102. How to achieve Database testing in Selenium?

As we all know Selenium WebDriver is a tool to automate User Interface. We could only interact with Browser using Selenium WebDriver.

Sometimes, we may face a situation to get the data from the Database or to modify (update/delete) the data from the Database. If we plan to automate anything outside the vicinity of a browser, then we need to use other tools to achieve our task. To achieve the Database connection and work on it, we need to use JDBC API Driver.

The Java Database Connectivity (JDBC) API provides universal data access from the Java programming language. Using the JDBC API, you can access virtually any data source, from relational databases to spreadsheets and flat files. It lets the user connect and interact with the Database and fetch the data based on the queries we use in the automation script. JDBC is a SQL level API that allows us to execute SQL statements. It creates a connectivity between Java Programming Language and the database.

Using JDBC Driver we could do the following

i. Establish a Database connection
ii. Send SQL Queries to the Database
iii. Process the results

## 103. How to delete Browser Cookies with Selenium Web Driver?

```
1 driver.Manage().Cookies.DeleteAllCookies();
```

TestNG – Interview Questions:

Here we have dealt with some important TestNG interview questions. If you want to learn more interview questions related to TestNG then here you go. We have a special post on TestNG Interview Questions. Also, you could find TestNG Complete Tutorial here

## 104. What is TestNG?

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing.

## 105. What are the annotations available in TestNG?

@BeforeTest
@AfterTest
@BeforeClass
@AfterClass
@BeforeMethod
@AfterMethod
@BeforeSuite
@AfterSuite
@BeforeGroups
@AfterGroups
@Test

## 106. What is TestNG Assert and list out some common Assertions supported by TestNG?

TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Some of the common assertions supported by TestNG are

- assertEqual(String actual,String expected)
- assertEqual(String actual,String expected, String message)
- assertEquals(boolean actual,boolean expected)
- assertTrue(condition)
- assertTrue(condition, message)
- assertFalse(condition)

- assertFalse(condition, message)

For Complete Post

## 107. How to create and run TestNG.xml?

In TestNG framework, we need to create T**estNG XML** file to create and handle multiple test classes. We do configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc in the XML file.

For Complete Post

## 108. How to set test case priority in TestNG?

We use *priority* attribute to the *@Test* annotations. In case priority is not set then the test scripts execute in alphabetical order.

```
1  package TestNG;

2  import org.testng.annotations.*;

3  public class PriorityTestCase{

4   @Test(priority=0)

5   public void testCase1() {

6   system.out.println("Test Case 1");

7   }

8   @Test(priority=1)

9   public void testCase2() {

10  system.out.println("Test Case 2");

11  }

12 }
```

Output:

1 Test Case 1

2 Test Case 2

## 109. What is Parameterized testing in TestNG?

*Parameterized tests* allow developers to run the same test over and over again using different values.

There are two ways to set these parameters:

- *with* `testng.xml` – <u>Practical Example</u>
- *with Data Providers* – <u>Practical Example</u>

## 110. How to run a group of test cases using TestNG?

TestNG allows you to perform sophisticated groupings of test methods. Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups (or regular expressions) while excluding another set.  This gives you maximum flexibility in how you partition your tests and doesn't require you to recompile anything if you want to run two different sets of tests back to back.

Groups are specified in your `testng.xml` file and can be found either under the `<test>` or `<suite>` tag. Groups specified in the `<suite>` tag apply to all the `<test>`tags underneath.

1 @Test (groups = { "smokeTest", "functionalTest" })

2 public void loginTest(){

3 System.out.println("Logged in successfully");

4 }

<u>View Complete Post</u>

## 111. What is the use of @Listener annotation in TestNG?

Ans. TestNG listeners are used to configure reports and logging. One of the most widely used listeners in TestNG is *ITestListener* interface. It has

methods like *onTestStart, onTestSuccess*, *onTestFailure*, *onTestSkipped* etc. We should implement this interface creating a listener class of our own. Next, we should add the listeners annotation (*@Listeners*) in the Class which was created.

Practical Example

## 112. How can we create a data-driven framework using TestNG?

By using *@DataProvider* annotation, we can create a Data Driven Testing Framework.

```
1  @DataProvider(name="getData")

2  public Object[][] getData(){

3  //Object [][] data = new Object [rowCount][colCount];

4  Object [][] data = new Object [2][2];

5

6  data [0][0] = "FirstUid";

7  data [0][1] = "FirstPWD";

8

9  data[1][0] = "SecondUid";

10  data[1][1] = "SecondPWD";

11

12  return data;

13

14  }
```

## 113. Where you have applied OOPS in Automation Framework?
## 114. How to handle browser (chrome) notifications in Selenium?
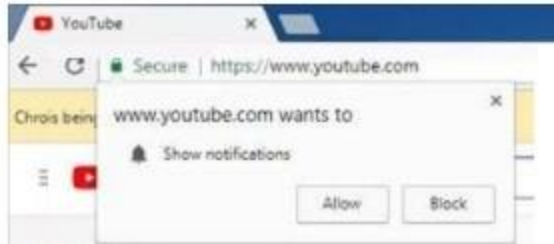
In Chrome, we can use ChromeOptions as shown below.

1 ChromeOptions options = new ChromeOptions();

2 options.addArguments("disable-infobars");

3 WebDriver player = new ChromeDriver(options);