



InterviewBit

Selenium Interview Questions



To view the live version of the page, [click here](#).

© Copyright by Interviewbit

Contents

Selenium Interview Questions for Freshers

1. What is Selenium?
2. What are the components of Selenium suite?
3. What is automation testing, and what are its advantages?
4. What are the advantages of using Selenium as an automation tool?
5. What are the disadvantages of using Selenium as a testing tool?
6. Why should Selenium be selected as a testing tool for web applications or systems?
7. What is Selenese? Explain different types of Selenium commands.
8. What is meant by a locator and name a few different types of locators present in Selenium.
9. State the major difference between “assert” and “verify” commands in Selenium.
10. What is exception test in Selenium?
11. What is XPath in Selenium. Explain XPath Absolute and XPath Relative.
12. In Xpath, what is the difference between "/" and "//"?
13. What is the difference between the commands "type" and "typeAndWait" in the context of Selenium?
14. Differentiate between findElement() and findElements() in the context of Selenium with proper examples.
15. In Selenium, how will you wait until a web page has been loaded completely?
16. What is Selenium WebDriver?
17. Is Selenium WebDriver a library?
18. Is Selenium WebDriver an interface or a class?
19. What are the different types of WebDriver Application Programming Interfaces in Selenium?
20. What programming languages does Selenium WebDiver support?

Selenium Interview Questions for Freshers

(.....Continued)

21. What open-source frameworks does Selenium WebDriver support?
22. What is WebDriver's super interface?
23. Explain the following line of code.
24. Is it necessary to use Selenium Server to run Selenium WebDriver scripts?
25. What will happen if I execute this command?
`driver.get("www.interviewbit.com");`
26. What is an alternative option to `driver.get()` method to open an URL in Selenium Web Driver?
27. What is the difference between `driver.get()` and `driver.navigate.to("url")`?
28. In Selenium WebDriver, what is the difference between `driver.getWindowHandle()` and `driver.getWindowHandles()`?
29. What are the differences between the methods `driver.close()` and `driver.quit()`?
30. What is the difference between `driver.findElement()` and `driver.findElements()` commands?
31. What are some cases that Selenium WebDriver cannot automate?
32. In Selenium WebDriver, what is an Object Repository?

Selenium Interview Questions for Experienced

33. Explain the difference between `driver.close()` and `driver.quit()` command in Selenium?
34. Explain the various navigation commands supported by Selenium?
35. Explain the same-origin policy and how Selenium handles it?
36. Explain the pause feature in Selenium IDE.
37. With the help of code snippets, explain how we can create right-click and mouse hover actions in Selenium.
38. Can we handle a windows-based pop-up in Selenium, and if not, then what are the alternatives?

Selenium Interview Questions for Experienced

(.....Continued)

39. Can you capture a screenshot using Selenium? If yes, write a simple code to illustrate the same.
40. Explain different types of framework and connection of Selenium with Robot Framework.
41. Demonstrate usage of Selenium through a test application.
42. Explain basic steps of Selenium testing and its widely used commands via a practical application.
43. What do you understand about the Page Object Model in the context of Selenium? What are its advantages?
44. What is Jenkins and what are the benefits of using it with Selenium?
45. How will you select a date from a datepicker in a webpage using Selenium for automated testing? Explain with a proper code.
46. What do you understand about broken links? How can you detect broken links in Selenium? Explain properly with code.
47. What do you understand about window handle in the context of automated testing? How can you handle multiple windows in Selenium?

Selenium WebDriver Interview Questions

48. How to create an Object Repository in your project?
49. What are the different types of waits that WebDriver supports?
50. Mention the several types of navigation commands that can be used?
51. How does a Selenium WebDriver interact with the browser?
52. What are Selenium WebDriver Listeners?
53. What is the implementation of WebDriver Listeners?
54. In Selenium WebDriver, how do you handle Ajax calls?
55. Which implementation of WebDriver promises to be the fastest?
56. At a bare minimum, how many parameters do selenium commands have?

Selenium WebDriver Interview Questions (.....Continued)

57. As seen below, we establish a WebDriver reference variable called 'driver.' What exactly is the purpose of proceeding in this manner?
58. What kinds of Selenium WebDriver exceptions have you run into?
59. What is the best way to deal with StaleElementReferenceException?
60. In a Selenium script, what happens if you use both implicit and explicit wait?
61. In a Selenium Script, what happens if you use both Thread.Sleep and WebDriver Waits?
62. In Selenium WebDriver, how do you take a screenshot?
63. How do I use Selenium WebDriver to enter text into a text box?
64. How can I type text into the text box without using the sendKeys() function?
65. How can I use Selenium WebDriver to clear the text in a text box?
66. What is the best way to acquire the textual matter of a web element?
67. How do I retrieve the value of an attribute in Selenium WebDriver?
68. How can I use Selenium WebDriver for clicking on a hyperlink?
69. How do I use Selenium WebDriver for submitting a form?
70. In Selenium WebDriver, how do I push the ENTER key on a text box?
71. How can I use WebDriver to mouse hover over a web element?
72. How does Selenium WebDriver handle hidden elements?
73. In Selenium WebDriver, how do you use the Recovery Scenario?
74. What is the Selenium WebDriver Architecture?

Selenium Tricky Interview Questions

Selenium Tricky Interview Questions

(.....Continued)

75. Why is it important to use TestNG when working with Selenium RC?
76. Can Selenium be used to test responsive web design?
77. What API should be used to test databases when using Selenium WebDriver for database testing?
78. What are assertions in Selenium?
79. What is the Silk Test Tool?
80. What is the purpose of the testing.xml file ?
81. What are the areas where Selenium can improve its features?
82. Can you explain what Page Factory is?
83. What is the Actions class?
84. What are the steps for troubleshooting tests using Selenium IDE?
85. What steps can be taken to resolve an issue where a Selenium script only works properly on Google Chrome but not Internet Explorer?
86. Is it possible to open pop-up windows with Selenium?
87. Can selenium be used to launch web browsers?
88. Is it possible to use only perform() without build()?
89. What is StaleElementReferenceException? When does this occur? And how to overcome such exceptions?
90. What are test design techniques? Explain BVA and ECP with some examples.

Let's get Started

Introduction

[Selenium](#) is a widely used open-source tool for automating web browsers. It allows developers to create scripts that can interact with web pages in a similar way to how a human user would. This makes it an invaluable tool for testing web applications and automating repetitive tasks.

In this article, we will be discussing a range of **Selenium Interview Questions and Answers**. These questions are designed to test your knowledge of Selenium, as well as your ability to think critically and solve problems. We will cover topics such as Selenium components, common challenges faced while using Selenium, and best practices for using Selenium in real-world scenarios.

Whether you are an experienced Selenium developer or just starting out, this article will provide valuable insights and tips to help you succeed in your next Selenium interview. So let's dive in and explore some of the most common Selenium interview questions and answers which are categorised in the following sections:

- [Selenium Interview Questions for Freshers](#)
- [Selenium Interview Questions for Experienced](#)
- [Selenium WebDriver Interview Questions](#)
- [Selenium Tricky Interview Questions](#)
- [Selenium MCQ Questions](#)

Selenium Interview Questions for Freshers

1. What is Selenium?

Selenium is an open-source (free) automated testing framework for validating web applications across multiple browsers and platforms. Selenium Test Scripts can be written in a variety of programming languages, including Java, C#, Python, and others. Selenium Testing is the term for testing done using the Selenium testing tool.

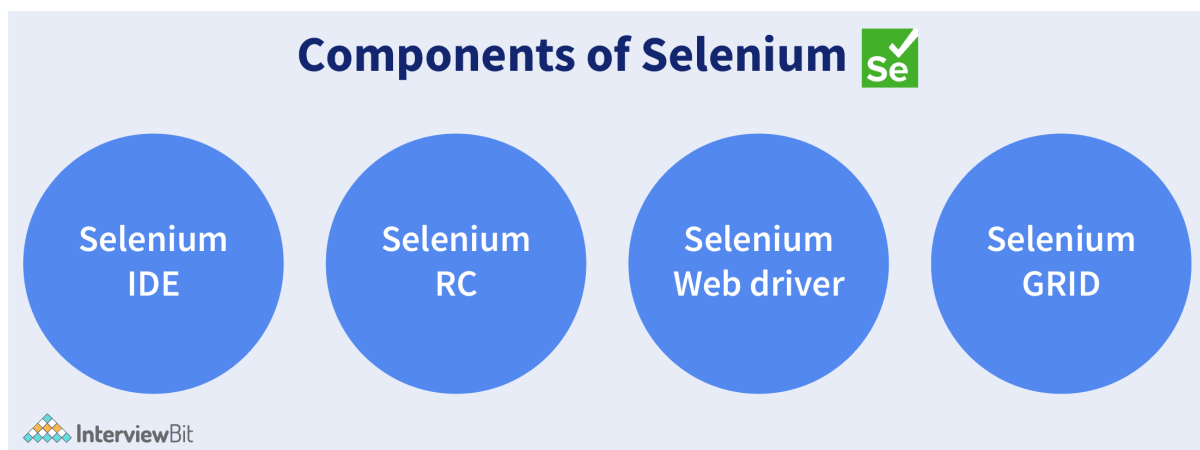
Selenium Software is a collection of tools, each of which caters to a specific organization's Selenium QA testing requirements. The following is a list of tools incorporated within Selenium:

- Selenium Integrated Development Environment (IDE).
- Selenium Remote Control (RC).
- Selenium WebDriver.
- Selenium Grid.

2. What are the components of Selenium suite?

Selenium is a package of several testing tools. It is therefore often referred to as a Selenium Suite with each of these tools designed to cater to a different testing requirement.

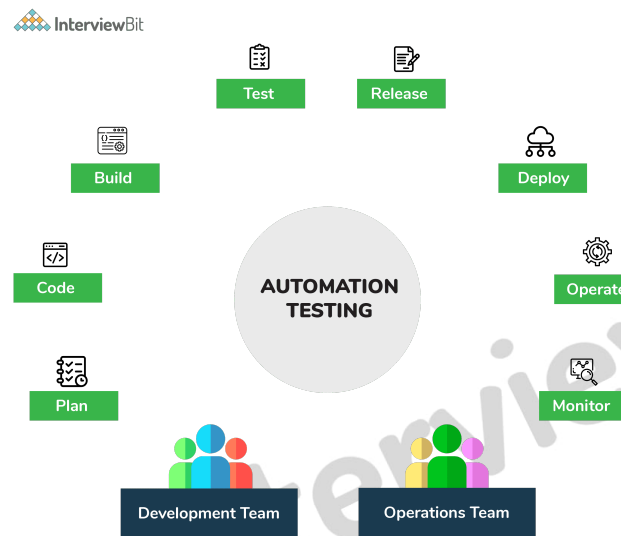
The following are the different components of Selenium Suite:



- **Selenium Integrated Development Environment (IDE):** It is a Firefox/Chrome plug-in that is developed to speed up the creation of automation scripts by recording the user actions on the web browser and exporting them as a reusable script.
- **Selenium Remote Control (RC):** It is a server that enables users to generate test scripts in their preferred programming language. It accepts commands from the test scripts and sends them to the browser as Selenium core JavaScript commands, for the browser to behave accordingly.
- **Selenium WebDriver:** It is a programming interface that helps create and run test cases by directly communicating with the web browser and using its native compatibility to automate. Unlike RC, it doesn't require an additional server to create and run test cases.
- **Selenium Grid:** It allows parallel execution of tests on different browsers and operating systems by distributing commands to other machines simultaneously.

3. What is automation testing, and what are its advantages?

Automation Testing or Test Automation is a process of automating the manual testing process of an application or a system by using testing tools that allow you to create scripts that can be executed repeatedly, generating detailed test reports of the application or system under test.



The advantages of Automated Testing are:

- It supports both the performance and functional testing of an application or system.
- It facilitates the execution of repeated test cases.
- It allows the parallel execution of the test cases.
- It improves the accuracy and efficiency of the system by reducing the manual intervention of humans to generate test cases.
- It helps in testing a large-scale test matrix.
- It saves valuable time and money for the testing team involved in the project.

4. What are the advantages of using Selenium as an automation tool?

The following are the advantages of using [Selenium for automated testing](#) :

- **Open-Source:** Selenium's greatest strength, as previously said, is that it is a freeware and portable tool. There are no out-of-pocket expenses. The utility can be downloaded for free, and community-based help is also accessible.
- **Language assistance:** Java, Perl, Python, C#, Ruby, Groovy, JavaScript, and other languages are supported by Selenium. It has its own script, yet it is not constrained by it. It can work with a variety of languages, depending on the developers' and testers' preferences.
- **Compatible with a variety of operating systems:** Selenium may run on a variety of operating systems, including Windows, Mac OS X, Linux, and UNIX. A customized testing suite can be constructed on any platform and then executed on another using the Selenium suite of products. For example, you may write test cases in Windows and run them on a Linux system with ease.
- **Browser compatibility:** Selenium is compatible with a variety of web browsers, including Internet Explorer, Chrome, Firefox, Opera, and Safari. When running tests and testing them across multiple browsers at the same time, this becomes really useful.
- **Programming languages and framework support:** Selenium works with a variety of programming languages and frameworks. For source code compilation, it can, for example, integrate with ANT or Maven frameworks. It may also be used to test apps and generate reports using the TestNG framework. Continuous Integration (CI), can integrate with Jenkins or Hudson, and it can also integrate with other open-source tools to offer other functionalities.
- **Tests on a variety of devices:** On Android, iPhone, and Blackberry, Selenium Test Automation can be used to automate mobile web application testing. This can aid in the generation of necessary results and the ongoing resolution of bugs present in the application.
- **Regular updates:** Selenium support is based on a community, which allows for frequent updates and upgrades. These upgrades are simple to install and don't require any special training. Selenium is thus both resourceful and cost-effective.
- **Selenium suites with a lot of content:** Selenium is more than just a single tool or utility; it's a full set of numerous testing tools that's why it's called a Suite. Each tool is tailored to specific testing needs and test environment constraints. Selenium also includes features such as Selenium IDE, Selenium Grid, and Selenium Remote Control (RC).
- **Ease with which it can be implemented:** Selenium has a user-friendly interface that makes it simple to develop and perform tests. Its open-source capabilities

5. What are the disadvantages of using Selenium as a testing tool?

The following are the disadvantages of using Selenium as a testing tool:

- **Tests web applications only:** Selenium supports the testing of only web-based applications. Mobile applications, Captcha, and Barcode readers cannot be tested using Selenium unless integrated with third-party tools like Appium and TestNG.
- **No built-in reporting and test management facility:** Selenium can generate reports only using third-party tools like TestNG or JUnit.
- **Unavailability of reliable tech support:** Since Selenium is an open-source tool, no dedicated support for user issues is available.
- **May require the knowledge of programming languages:** Some prior programming knowledge is required to use Selenium.

6. Why should Selenium be selected as a testing tool for web applications or systems?

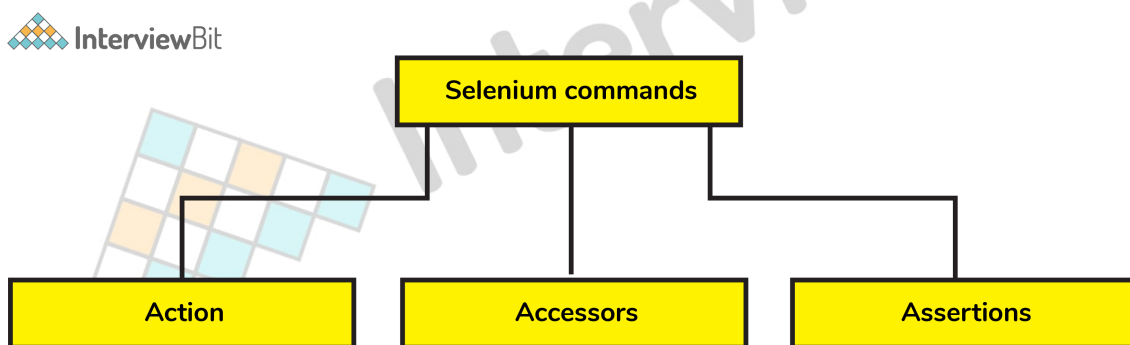
Selenium provides the following advantages, which make it an excellent automated testing framework:

- It is free and open-source software with a large user base and supports providing community.
- It has cross-browser compatibility and supports multiple browsers like Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It supports multiple operating systems such as Windows, Linux, macOS, etc.
- It facilitates the usage of multiple programming languages including Scala, Ruby, Python, PHP, Perl, Java, Groovy, C#, etc.
- It provides support for distributed testing as well.

7. What is Selenese? Explain different types of Selenium commands.

The language used for writing test scripts in Selenium IDE is called Selenese. It is a set of commands used to test your web application or system. Selenium commands could be divided into 3 major categories:

- **Actions:** These are the commands interacting directly with web applications.
- **Accessors:** These are the commands which allow users to store values in a user-defined variable.
- **Assertions:** They enable a comparison of the current state of the application with its expected state.



8. What is meant by a locator and name a few different types of locators present in Selenium.

A locator is an address for uniquely identifying web elements within a web page. There are different types of locators present in Selenium to identify web elements uniquely and accurately like:

- ID
- ClassName
- Name
- TagName
- LinkText
- PartialLinkText
- Xpath
- CSS Selector
- DOM.

9. State the major difference between “assert” and “verify” commands in Selenium.

Both “assert” and “verify” commands check whether the given condition is true or false and the only difference between them is that:

- **Assert:** Assert condition stops the execution of the testing if the given condition is false else would continue with the further tests.
- **Verify:** Verify the condition doesn't stop the flow of execution irrespective of the condition being true or false.

10. What is exception test in Selenium?

An exception test is a test that expects an exception to be thrown inside a test class. It expects a `@Test` annotation followed by the expected exception name in the brackets.

Eg: `@Test(expectedException = NoSuchElementException.class)` is an exception test for missing elements in Selenium.

11. What is XPath in Selenium. Explain XPath Absolute and XPath Relative.

XPath, also defined as XML-Path (Extensible Markup Language Path), is a language used to query XML documents and provide functionalities like locating elements in Selenium by iterating through each element in a webpage. In XPath, data is stored in a key-value pair format similar to an HTML tag. It uses a single slash, i.e. ' / ' for creating an absolute path, and a double slash, i.e. ' // ' for creating a relative path for an element to be located on a webpage.

12. In Xpath, what is the difference between "/" and "//"?

- **Single Slash "/"** - A single slash is used to create an Xpath with an absolute path, i.e., the XPath will begin with the document node/start node. For example,

```
Absolute XPath: /html/body/div/div/form/input
```

Here, /html is the root html node.

- **Double Slash "//"** - The double slash is used to construct an Xpath with a relative path, which means the XPath can start selection from anywhere on the page. For example,

```
Relative XPath: //input[@id = 'email']
```

Here, we can locate an input having id = 'email' present anywhere in the document object model (DOM).

13. What is the difference between the commands "type" and "typeAndWait" in the context of Selenium?

The "type" command is used to enter keyboard key values into a software web application's text box. It can also be used to choose values from a combo box, whereas the "typeAndWait" command is used when you finish typing and the software web page begins to reload. This command will wait for the page of the software program to reload before proceeding. You must use a basic "type" command if there is no page reload event when typing.

14. Differentiate between `findElement()` and `findElements()` in the context of Selenium with proper examples.

Following table lists the differences between `findElement()` and `findElements()` in Selenium:

<code>findElement()</code>	<code>findElements()</code>
The first web element that matches the locator is returned.	It gives you a list of all the web items that match the locator.
If there are no matching web elements, a <code>NoSuchElementException</code> is produced.	If there are no matching elements, an empty list is returned.
Syntax – <code>WebElement button = webdriver.findElement(By.name("<<Name value>>"));</code>	Syntax – <code>List<WebElement> buttons = webdriver.findElements(By.name("<<Name value>>"));</code>

- Using `findElements()` :-


```
// JAVA
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;
public class findElements {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\vaibhav\\Desktop\\Java\\");
        WebDriver driver = new ChromeDriver();
        String url = "https://www.exampleurl.com/example.htm";
        driver.get(url);
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

        List<WebElement> rows = driver.findElements(By.xpath("//table/tbody/tr[2]/td"));
        System.out.println("The number of values in row 2 is "+ rows.size());
        driver.close();
    }
}
```

Explanation - In the above code, first of all, we import all the necessary headers and then set up the driver for the chrome browser. We use the `findElements()` method to find all the values present in the 2nd row of a table in the given URL web page using the XPath of the element.

- Using `findElement()` :-

```
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;
public class findTagName {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\vaibhav\\Desktop\\Java\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        String url = "https://www.exampleurl.com/example.htm";
        driver.get(url);
        driver.manage().timeouts().implicitlyWait(12, TimeUnit.SECONDS);
        driver.findElement(By.cssSelector("input[id='search']")).sendKeys("Selenium"); //WebElement element = driver.findElement(By.cssSelector("input[id='search']"));
        driver.close();
    }
}
```

Explanation - In the above code, first of all, we import all the necessary headers and then set up the driver for the chrome browser. We use the `findElement()` method to find an input element having an id attribute set as search.

15. In Selenium, how will you wait until a web page has been loaded completely?

There are two methods of making sure that the web page has been loaded completely in Selenium.

They are as follows:

1. Immediately after creating the webdriver instance, set an implicit wait:

```
temp = driver.Manage().Timeouts().ImplicitWait;
```

On every page navigation or reload, this will try to wait until the page is fully loaded.

2. Call JavaScript return `document.readyState` till "complete" is returned after page navigation. As a JavaScript executor, the web driver instance can be used.

Code example:

```
new WebDriverWait(firefoxDriver, pageLoadTimeout).until(  
    webDriver -> ((JavascriptExecutor) webDriver).executeScript("return document.readyState")
```

16. What is Selenium WebDriver?

Selenium WebDriver, also known as Selenium 2, is a browser automation framework that accepts and sends commands to a browser to implement it. It has direct control over the browser because it communicates with it directly. Java, C#, PHP, Python, Perl, and Ruby are all supported by Selenium WebDriver.

17. Is Selenium WebDriver a library?

Selenium WebDriver is a prominent free open-source library for automating browsers and testing web applications.

18. Is Selenium WebDriver an interface or a class?

Selenium WebDriver is usually a set of methods defined by an interface. The browser-specific classes, on the other hand, provide an implementation of it by extending a class. `AndroidDriver`, `ChromeDriver`, `FirefoxDriver`, `InternetExplorerDriver`, `SafariDriver`, and others are some of the implementation classes.

19. What are the different types of WebDriver Application Programming Interfaces in Selenium?

The Various **Types of WebDriver APIs** in Selenium are as follows:

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

20. What programming languages does Selenium WebDriver support?

The various programming languages that Selenium WebDriver supports are as follows:

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

21. What open-source frameworks does Selenium WebDriver support?

The following are the open-source frameworks supported by the Selenium WebDriver:

- **TestNG:**
 - Cédric Beust designed TestNG, a testing framework for the Java programming language that was influenced by JUnit and NUnit.
 - TestNG was created with the purpose of covering a wider range of test categories, including unit, functional, end-to-end, integration, and so on, with more robust and user-friendly functions.
- **JUnit:**
 - It is used for Unit Testing of various types of applications.

22. What is WebDriver's super interface?

SearchContext is the Super Interface of the WebDriver.

23. Explain the following line of code.

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

'WebDriver' is an interface, and we are generating a WebDriver object by instantiating a FirefoxDriver object (This object uses Firefox Driver to link the test cases with the Firefox browser).

24. Is it necessary to use Selenium Server to run Selenium WebDriver scripts?

Selenium Server is required when distributing Selenium WebDriver scripts for execution with Selenium Grid. Selenium Grid is a Selenium functionality that allows you to execute test cases on multiple machines on various platforms. You wish to execute your test cases on a remote machine because your local machine is running numerous applications. You will need to set up the remote server so that the test cases can run on it.

25. What will happen if I execute this command? `driver.get("www.interviewbit.com") ;`

An exception is triggered if the URL does not begin with http or https. As a result, the HTTP protocol must be sent to the `driver.get()` method.

26. What is an alternative option to `driver.get()` method to open an URL in Selenium Web Driver?

`driver.navigate()` can be used instead. It is used for navigating forwards and backwards in a browser.

27. What is the difference between `driver.get()` and `driver.navigate.to("url")`?

The difference between the two is as follows:

- `driver.get()` : To open a URL and have it wait for the entire page to load.
- `driver.navigate.to()` : To navigate to a URL without having to wait for the entire page to load.

28. In Selenium WebDriver, what is the difference between `driver.getWindowHandle()` and `driver.getWindowHandles()`?

The difference between the two is as follows:

- `driver.getWindowHandle()` – This method returns the current page's handle (a unique identifier).
- `driver.getWindowHandles()` – This function returns a list of handles for all of the pages that are opened at that particular time.

29. What are the differences between the methods `driver.close()` and `driver.quit()`?

The functions of these two methods (`driver.close` and `driver.quit`) are nearly identical. Although both allow us to close a browser, there is a distinction.

- To close the current WebDriver instance, use `driver.close()` .
- To close all open WebDriver instances, use `driver.quit()` .

30. What is the difference between `driver.findElement()` and `driver.findElements()` commands?

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

- `findElement()` - Based on the locator supplied as an argument, `findElement()` returns a single `WebElement` which is found first. If no element is discovered, it throws the `NoSuchElementException`.

`findElement()` has the following syntax:

```
WebElement textbox = driver.findElement(By.id("textBoxLocator"));
```

- `findElements()` - Based on the locator supplied as an argument, `findElements()` , on the other hand, produces a list of `WebElements` that all satisfy the location value supplied. If no element is discovered, it does not throw any exception and returns a list of zero elements.

`findElements()` has the following syntax:

```
List <WebElement> elements = element.findElements(By.id("value"));
```

31. What are some cases that Selenium WebDriver cannot automate?

Some of the scenarios which we cannot automate are as follows:

- Selenium WebDriver does not support bitmap comparison.
- Using Selenium WebDriver to automate Captcha is not possible.
- Using Selenium WebDriver, we are unable to read bar codes.
- Video streaming scenarios: Selenium will almost never be able to recognise video controllers. To some extent, JavaScript Executor and flex UI selenium will work, although they are not completely dependable.
- Performance testing can be automated, however, it's preferable to avoid using Selenium for performance testing.

32. In Selenium WebDriver, what is an Object Repository?

Instead of hard-coding element locator data in the scripts, the Object Repository is used to store the element locator data in a centralized location. To store all of the element locators, we create a property file (.properties), which acts as an object repository in Selenium WebDriver.

Selenium Interview Questions for Experienced

33. Explain the difference between `driver.close()` and `driver.quit()` command in Selenium?

Following is the major difference between both the commands:

- `driver.close()` command closes the currently active window on which the user is working or the window being currently accessed by the web driver.
- `driver.quit()` command, unlike the `driver.close()` command, closes all the windows opened by the program and hence should be used with care.

Both the commands don't take any parameter and don't return any value either.

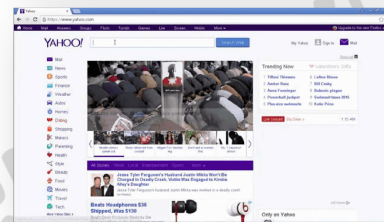
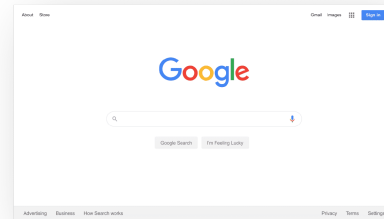
34. Explain the various navigation commands supported by Selenium?

Selenium has the support of majorly 4 navigation commands:

- `navigate().back()` : This command is used for taking the user to the last webpage of the browser history.
- `navigate().forward()` : This command is used for taking the user to the next web page of the browser history.
- `navigate().refresh()` : This command is used for reloading the web components of a webpage by refreshing it.
- `navigate().to()` : This command is used for navigating to a particular URL in a new web browser. It takes the URL to be migrated to, as a parameter.

35. Explain the same-origin policy and how Selenium handles it?

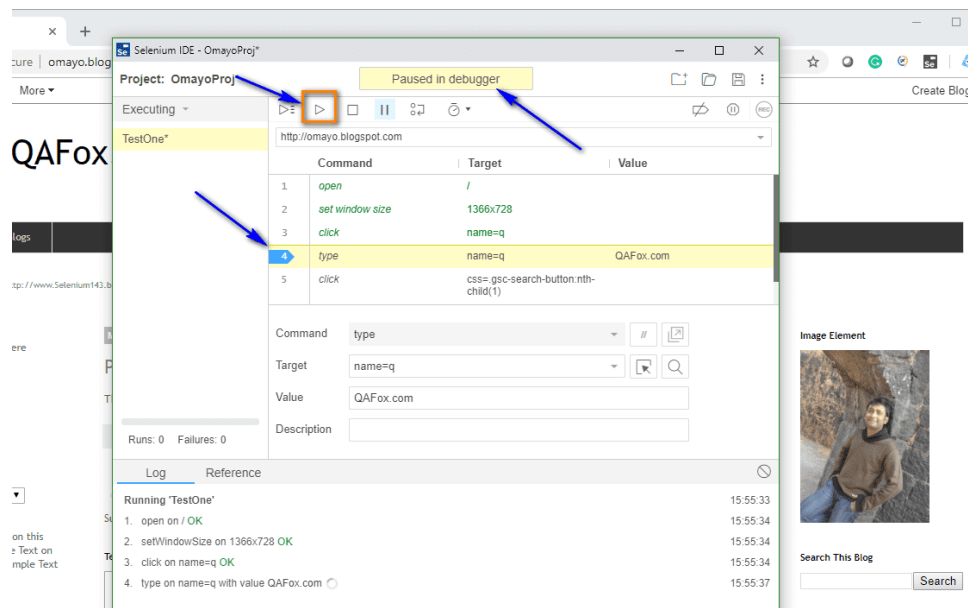
The same Origin policy is a feature adopted for security purposes that allows a web browser to run scripts from one webpage to access the contents of another webpage provided both pages have the same origin. The URL scheme, hostname, and port number combo are referred to as the origin. This policy was introduced to prevent access to sensitive data on one webpage by another for ill purposes. Consider a Java program used by `scaler.com`, the program can access domain pages like `scaler.com/mentors`, `scaler.com/courses` but none from different domains like `facebook.com`.



The Selenium Server (Selenium RC) acts as a client-configured HTTP proxy and "tricks" the browser into believing that Selenium Core and the web application being tested come from the same origin.

36. Explain the pause feature in Selenium IDE.

The pause feature is built to handle exceptions in the test script by allowing the user to pause at the statement causing the exception and enter the debug mode by clicking on the pause icon on the top right corner of the IDE. This feature prevents the entire test case's failure and gives the user a chance to correct the error instantly.



37. With the help of code snippets, explain how we can create right-click and mouse hover actions in Selenium.

The following code can replicate right-click action:

```
actions action = newActions(driver);
WebElement element = driver.findElement(By.id("elementId"));
action.contextClick(element).perform();
```

The following code can replicate the mouse hover action:

```
actions action = newActions(driver);
WebElement element = driver.findElement(By.id("elementId"));
action.moveToElement(element).perform();
```

38. Can we handle a windows-based pop-up in Selenium, and if not, then what are the alternatives?

No, Selenium doesn't support windows-based pop-ups as it's an automated testing tool built for web application-based testing. However, with the support of third-party tools like AutoIT, Robot class, etc., windows-based pop-ups can be handled in selenium.

39. Can you capture a screenshot using Selenium? If yes, write a simple code to illustrate the same.

Yes, using a web driver in Selenium, we can capture the screenshot. Following is the code to do the same:

```
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 TakeScreenshot {
    WebDriver drv;
    @ Before
    public void setUp() throws Exception {
        driver = new FirefoxDriver();
        drv.get("https://google.com");
    }
    @ After
    public void tearDown() throws Exception {
        drv.quit();
    }
    @ Test
    public void test() throws IOException {
        // Capture the screenshot
        File scrFile = ((TakeScreenshot)drv).getScreenshotAs(OutputType.FILE);
        // Code for pasting screenshot to a user-specified location
        FileUtils.copyFile(scrFile, new File("C:\\Screenshot\\Scr.jpg"))
    }
}
```

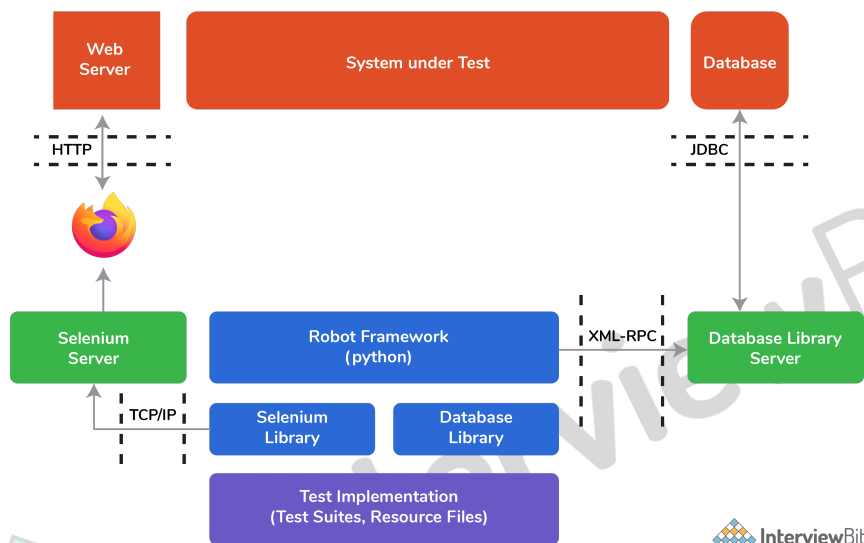
40. Explain different types of framework and connection of Selenium with Robot Framework.

The following are the different types of frameworks:

- **Behavior-Driven Development Framework:** This type of framework provides a readable and easily understandable format to Business Analysts, Developers, Testers, etc.
- **Data-Driven Testing Framework:** This type of framework helps separate test data from the test-script logic by storing test data in some external database in the form of key-value pairs. These keys can be used for accessing as well as populating data into the test scripts.
- **Keyword-Driven Testing Framework:** This type of framework is an extension of the data-driven testing framework. In addition to separating test data and the test-script logic, it also separates a part of the test script code by storing it in an external data file.
- **Library Architecture Testing Framework:** This type of framework groups common steps into functions under a library and calls these functions as and when required.
- **Module-Based Testing Framework:** This type of framework divides each test application into several isolated and logical modules, with each module having its distinct test script.
- **Hybrid Testing Framework:** This type of framework is a combination of the above-mentioned frameworks leveraging all their good features.

[Robot Framework](#) is a modular open-source automation framework that can interact with 3rd party libraries and functions. To execute a web testing library such as Selenium, a test automation runner or an automation wrapper is required, which is provided to it in the form of Robot Framework. Other popular test runners to serve the same purpose are MSTest, TestNG, Nunit, Junit, etc.

The below diagram shows the connection of the Robot framework to the Selenium library:



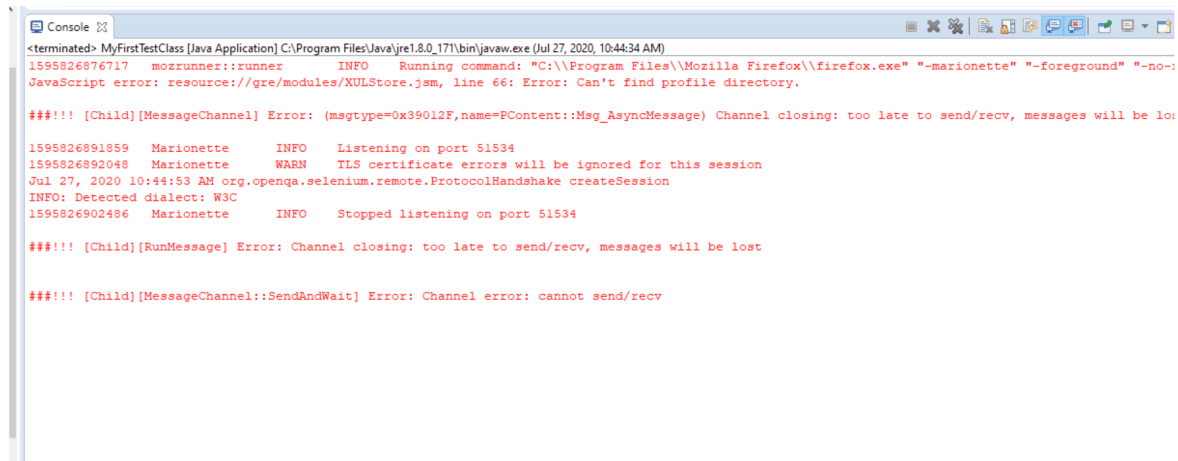
41. Demonstrate usage of Selenium through a test application.

You need the following prerequisites to run a demo Selenium test script:

- **Java SDK** in your respective Operating System.
- A **Java-based IDE** such as Eclipse or IntelliJ.
- A **Selenium WebDriver** is to be added as a dependency to Java IDE.

```
package scalerAcademy;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.WebDriver;
public class MyFirstTestClass {
public static void main(String[] args) throws InterruptedException {
//It sets the system property to the given value.
System.setProperty("webdriver.gecko.driver", "D:\\Softwares\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
    driver.get("https://www.google.com/");
    //Launch website in the browser
    driver.manage().window().maximize();
    //The sleep pauses the execution of the thread for 5000 ms.
    Thread.sleep(5000);
    driver.quit();
}
}
```

Once you run the above script in a Java IDE, you'll get the following execution logs displayed in your IDE window.



```
<terminated> MyFirstTestClass [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Jul 27, 2020, 10:44:34 AM)
1595826876717 mozrunner::runner INFO Running command: "C:\Program Files\Mozilla Firefox\firefox.exe" "--marionette" "--foreground" "--no-;
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile directory.

###!!! [Child][MessageChannel] Error: (msgtype=0x39012F,name=PContent::Msg_AsyncMessage) Channel closing: too late to send/recv, messages will be lo;

1595826891859 Marionette INFO Listening on port 51534
1595826892048 Marionette WARN TLS certificate errors will be ignored for this session
Jul 27, 2020 10:44:53 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
1595826902486 Marionette INFO Stopped listening on port 51534

###!!! [Child][RunMessage] Error: Channel closing: too late to send/recv, messages will be lost

###!!! [Child][MessageChannel::SendAndWait] Error: Channel error: cannot send/recv
```

42. Explain basic steps of Selenium testing and its widely used commands via a practical application.

Selenium testing can be divided into the following seven basic elements:

1. Creating an instance of a WebDriver: This is the first step for all the usages of a Selenium webdriver API. An instance of a webdriver interface is created using a constructor of a particular browser. This webdriver instance is used to invoke methods and access other interfaces. The following are the most commonly used commands for initializing a web driver:

```
Firefox:
WebDriver driver = new FirefoxDriver();
Chrome:
WebDriver driver = new ChromeDriver();
Safari Driver:
WebDriver driver = new SafariDriver();
Internet Explorer:
WebDriver driver = new InternetExplorerDriver();
```

2. Navigating to a webpage: The second step after initializing an instance of a webdriver, is to navigate to a particular webpage you want to test. The following are the most commonly used commands for webpage navigation:

```
Navigate to URL:
driver.get("https://www.interviewbit.com")
driver.navigate().to("https://www.interviewbit.com")
Refresh page:
driver.navigate().refresh()
Navigate forward in browser history:
driver.navigate().forward()
Navigate backward in browser history:
driver.navigate().backward()
```

3. Locating an HTML element on the webpage: To interact with a web element and perform actions on it like clicking a button or entering text, we first need to locate the desired elements such as the button or the textbox on the web page. The following are the most commonly used commands for web element navigation:

```
Locating by ID:
driver.findElement(By.id("q")).sendKeys("Selenium 3");
Location by Name:
driver.findElement(By.name("q")).sendKeys ("Selenium 3");
Location by Xpath:
driver.findElement(By.xpath("//input[@id=='q']")).sendKeys("Selenium 3");
Locating Hyperlinks by Link Text:
driver.findElement(By.LinkText("edit this page")).Click();
Locating by ClassName
driver.findElement(By.className("profileheader"));
Locating by TagName
driver.findElement(By.tagName("select')).click();
Locating by LinkText
driver.findElement(By.linkText("NextPage")).click();
Locating by PartialLinkText
driver.findElement(By.partialLinkText(" NextP")).click();
```

4. Performing actions on an HTML element: Once we have located the HTML element, the next step is interacting with it. The following are the most commonly used commands for performing actions on an HTML elements:

```
Entering a username
usernameElement.sendKeys("InterviewBit");
Entering a password
passwordElement.sendKeys("Raw");
Submitting a text input element
passwordElement.submit();
Submitting a form element:
formElement.submit();
```

5. Anticipating browser response from the action: Once an action is performed, anticipating a response from the browser to test comes under this step. It takes a second or two for the action to reach the browser, and hence wait is often required for this step. There are two main types of wait conditions:

Implicit Wait: It sets a fixed, definite time for all the webdriver interactions. It's slightly unreliable as web driver response times are usually unpredictable. Eg:

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Explicit Wait: This type of wait condition sets an expected condition to occur on the web page or a maximum wait time for all the webdriver interactions. Eg:

```
WebElement messageElement = wait.until(ExpectedConditions.presenceOfElementLocated(
```

6. Running tests and recording their results using a test framework: in this step, we run tests in an automated test script to evaluate an application's function and performance. Various test frameworks are used for this step, such as:

1. JUnit for Java
2. NUnit for C#
3. unittest or Pyunit for Python
4. RUnit for Ruby

Most frameworks use some sort of assert statement to verify their test results from the expected results. Eg:


```
assertEquals (expectedMessage, actualMessage);
```

7. Concluding a test: In this step, we conclude a test by invoking a quit method on the driver variable. This step closes all the webpages, quits the WebDriver server, and releases the driver. Eg:

```
driver.quit();
```

The following is an example of an app that covers all the steps mentioned above:

```
import org.openqa.selenium.By,
import org.openqa.selenium.WebElement,
import org.openqa.selenium.support.ui.ExpectedConditionOf, import org.openqa.selenium.support
import org.junit.Assert;
public class Example {
public static void main(String[] args) {
// Creating a driver instance
WebDriver driver = new FirefoxDriver(),
// Navigate to a web page
driver.get("http://www.foo.com");
// Enter text to submit the form
WebElement usernameElement = driver.findElement( By.name("username"));
WebElement passwordElement = driver.findElement(By.name("password"));
WebElement formElement = driver.findElement(By.id("loginForm"));
usernameElement.sendKeys("Scaler Academy");
passwordElement.sendKeys("Raw");
formElement.submit(); // submit by form element

//Putting an explicit wait
WebDriverWait wait = new WebDriverWait(driver, 10);
WebElement messageElement = wait.until(
    ExpectedConditions.presenceOfElementLocated(By.id("loginResponse"))
);
// Run a test
String message = messageElement.getText();
String successMsg = "Welcome to foo. You logged in successfully.";
Assert.assertEquals (message, successMsg);
// Conclude a test
driver.quit();
}
}
```

43. What do you understand about the Page Object Model in the context of Selenium? What are its advantages?

Page Object Model (POM) is a design pattern that generates an Object Repository for web UI elements and is widely used in test automation. The paradigm has the advantage of reducing code duplication and improving test maintenance. According to this paradigm, each web page in the application should have its own Page Class. This Page class will identify the web page's WebElements and also has Page methods that operate on those WebElements. The names of these methods should correspond to the tasks they perform, for example, if a loader is waiting for the payment gateway to appear, the POM method name could be

```
waitForPaymentScreenDisplay() .
```

The following are the advantages of the Page Object Model (POM) :

- According to the Page Object Design Pattern, user interface activities and flows should be separated from verification. Our code is clearer and easier to understand as a result of this notion.
- The second advantage is that the object repository is independent of test cases, allowing us to reuse the same object repository with different tools. For example, we can use Selenium to combine Page Object Model with TestNG/JUnit for functional testing and JBehave/Cucumber for acceptability testing.
- Because of the reusable page methods in the POM classes, code gets less and more efficient.
- Methods are given more realistic names that can be easily associated with the UI operation. If we land on the home page after clicking the button, the function name will be 'gotoHomePage()'.

44. What is Jenkins and what are the benefits of using it with Selenium?

Hudson Lab's Jenkins is the most popular open-source continuous integration technology. It's cross-platform, meaning it can run on Windows, Linux, Mac OS, and Solaris. Jenkins is a Java application. Jenkin's main purpose is to keep track of any job, such as SVN (Apache Subversion) checkouts, cron jobs, or application states. When a specific event in an application occurs, it triggers pre-configured actions.

[Learn More.](#)



The following are the **features of Jenkins**:

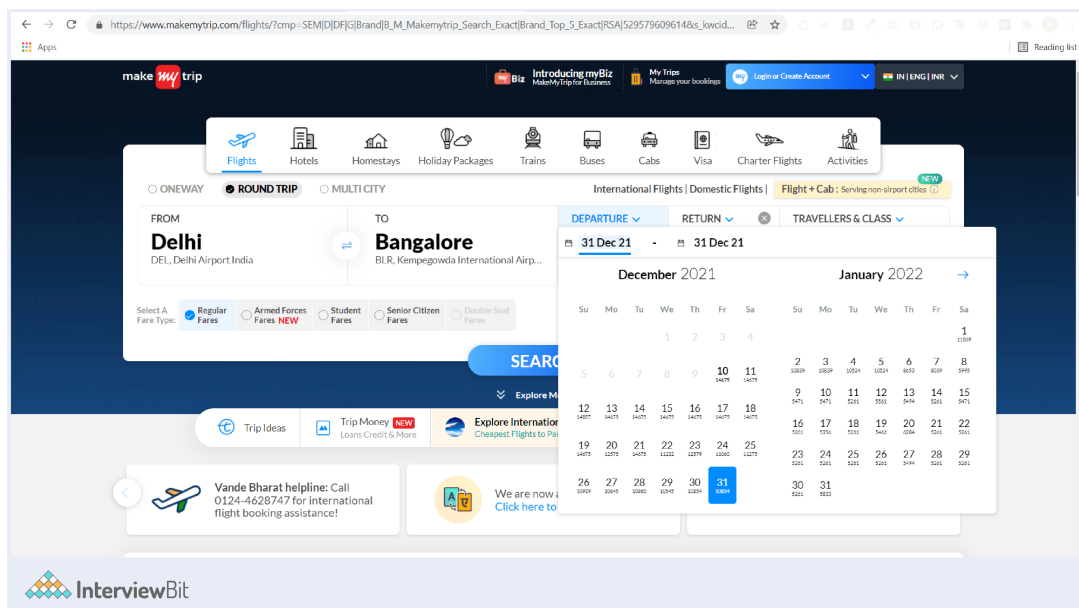
- Jenkins generates a list of all changes made in SVN repositories, for example.
- Jenkins gives permanent links to the most recent build or failed build, which can be utilized for convenient communication.
- Jenkins is simple to install using either a direct installation file (exe) or a war file for deployment via the application server.
- Jenkins can be set up to send the content of the build status to an email address.
- Simple Configuration: Jenkins makes it simple to set up multiple tasks.
- Jenkins can be configured to run the automated test build on TestNg following every SVN build.
- Jenkins documents the details of the jar, its version, and the mapping of build and jar numbers.
- Plugins: Jenkins can be set to utilize features and additional functionality provided by third-party plugins.

Following are the reasons we **use Jenkins with Selenium**:

- When you run Selenium tests in Jenkins, you can run them every time your program changes, and when the tests pass, you may deploy the software to a new environment.
- Jenkins may execute your tests at a predetermined time.
- The execution history as well as the Test Reports can be saved.
- Jenkins allows you to develop and test a project in continuous integration using Maven.

45. How will you select a date from a datepicker in a webpage using Selenium for automated testing? Explain with a proper code.

In such types of questions, the interviewer wants to assess how clear your understanding is of the framework. It is a good practice to explain the code while you write it so that the interviewer is engaged at all points and does not feel left out. We will be considering an example on MakeMyTrip.



Here, we will be using the chrome browser and so we will be implementing the code for the chrome browser only. You can implement similar code for firefox and other browsers as well.

First of all, we create a package named browserSelection which contains a class defined for handling different types of browsers such as chrome, firefox that we may want to use.

```
package browserSelection;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class SelectBrowser
{
    static WebDriver driver;
    public static WebDriver useChrome()
    {
        System.setProperty("webdriver.chrome.driver", "E:\\SeleniumLibs\\chromedriver_wi
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        return driver;
    }
}
```

Next, we create a package named datepicker which will contain a class containing methods defined for selecting a specific date on the website of MakeMyTrip. We need to import this package into our driver class and call the respective methods.



```
package datepicker;
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.WebElement;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import browserSelection.SelectBrowser;
public class DatePick
{
    WebDriver driver;
    @BeforeMethod
    public void startBrowser()
    {
        driver = SelectBrowser.useChrome();
    }
    @Test
    public void selectDateUtil() throws InterruptedException, AWTException
    {
        //Modify Wait time as per the Network Ability in the Thread Sleep method
        driver.get("https://www.makemytrip.com/");
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        Thread.sleep(5000);
        try
        {
            driver.findElement(By.xpath("//input[@id='hp-widget__depart']")).click();
            Thread.sleep(2000);
            Date sampleDate = new Date(); // initialising the date object with the current
            SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM yyyy");
            String date = formatter.format(sampleDate); // formatting the date object in dd-MMM-yyyy
            String splitter[] = date.split("-");
            String monthYear = splitter[1]; // storing the month and year concatenated string
            String day = splitter[0]; // storing the day number in the current date
            System.out.println(monthYear);
            System.out.println(day);

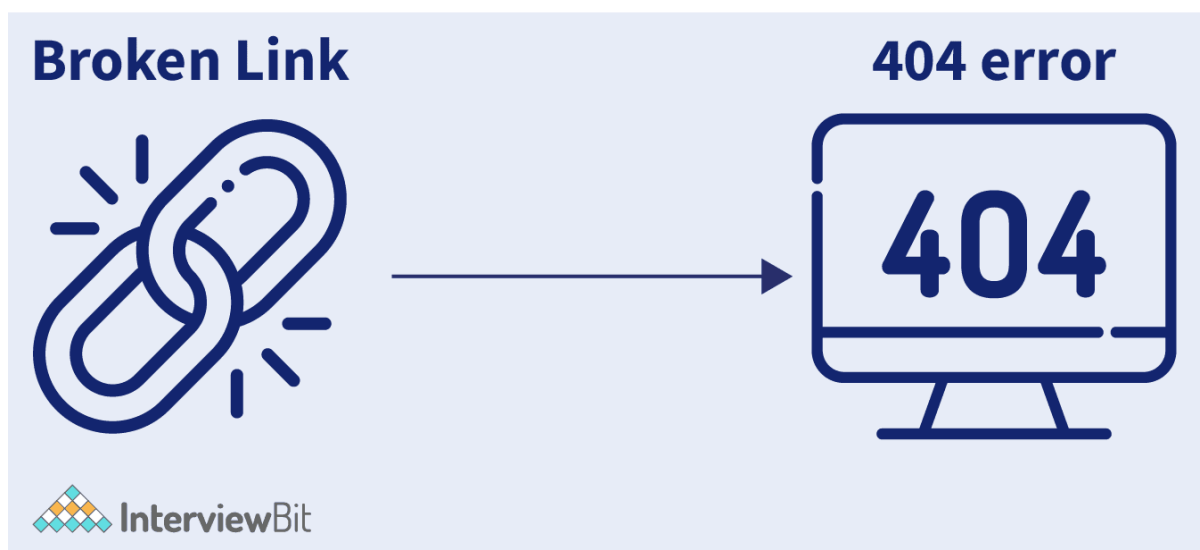
            selectDate(monthYear, day); // function invocation
            Thread.sleep(3000);

            public void selectDate(String monthYear, String select_day) throws InterruptedException
            {
                List<WebElement> elements = driver.findElements(By.xpath("//div[@class='ui-c
                for (int i=0; i<elements.size();i++)
                {
                    System.out.println(elements.get(i).getText());
                    //Selecting the month
                    if(elements.get(i).getText().equals(monthYear))
                    {
                        //Selecting the date
                        List<WebElement> days = driver.findElements(By.xpath("//div[@class='ui-
```

In the above code, the function `startBrowser()` is used to invoke the `useChrome()` method from the imported package `browserSelection`. The function `selectDateUtil()` is used to select the current date from the date picker of the sample web page. The `endBrowser()` function is used to close the driver connections by invoking the `quit()` method.

46. What do you understand about broken links? How can you detect broken links in Selenium? Explain properly with code.

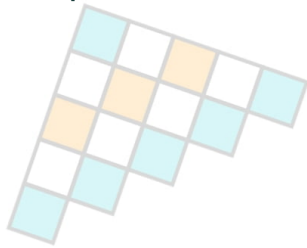
Links or URLs that are not reachable are known as broken links. They may be unavailable or inoperable due to a server issue. A URL's status will always be 2xx, indicating that it is legitimate. There are a variety of HTTP status codes, each with its own set of functions. HTTP status 4xx and 5xx indicate an invalid request. The 4xx class of status codes is used for client-side errors, while the 5xx class is used for server response errors.



You should always check for broken links on your site to ensure that the user does not end up on an error page. If the rules aren't updated appropriately, or the requested resources aren't available on the server, the error will occur. Manual link checking is a time-consuming task because each web page may have a huge number of links, and the process must be performed for each page.

To find broken links in Selenium, follow the instructions below.

- Using the <a> (anchor) tag, collect all of the links on a web page.
- For each link, send an HTTP request.
- Make that the HTTP response code is correct.
- Based on the HTTP response code, determine whether the link is genuine or not.
- Repeat the procedure for all of the links that were captured in the first step.



```
package SeleniumPackage;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Iterator;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
public class BrokenLinks {

    public static void main(String[] args) {

        String pageURL = "http://www.interviewbit.com";
        String url = "";
        HttpURLConnection huc = null;
        int responseCode = 200;
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\user\\Downloads\\seler");
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1200","-");
        WebDriver driver = new ChromeDriver(options); //Creating an instance of the WebDr

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

        driver.get(pageURL);

        List<WebElement> links = driver.findElements(By.tagName("a")); // getting hold of all the links

        Iterator<WebElement> it = links.iterator();
        // Iterating over the obtained list of elements and checking them one by one
        while(it.hasNext()){

            url = it.next().getAttribute("href");

            System.out.println(url);

            if(url == null || url.isEmpty()){
                System.out.println("The linked element has invalid href url.");
                continue;
            }

            if(!url.startsWith(pageURL)){
                System.out.println("URL belongs to another domain, skipping it.");
                continue;
            }

            try {
                huc = (HttpURLConnection)(new URL(url).openConnection());

                huc.setRequestMethod("HEAD");

                huc.connect(); // connecting to the url

                responseCode = huc.getResponseCode(); // reading the response code on file
```

Explanation - In the above code, we first set up the system properties and then initialize a webdriver object. We find all the elements in the web page having the anchor tag with the help of the `findElements()` method. Then, we iterate over the list obtained one by one and fire up the URL and read the response code received to check if it is a broken link or not.

47. What do you understand about window handle in the context of automated testing? How can you handle multiple windows in Selenium?

The window handle is a one-of-a-kind identifier that contains the addresses of all of the windows. Consider it a window pointer that returns the string value. Each browser will presumably have its own window handle. This window handle function aids in the retrieval of all window handles.

Syntax:

- `get.windowhandle()` : This function is used to retrieve the current window's handle.
- `get.windowhandles()` : This function is useful for retrieving the handles of all the windows that have been opened.
- `set`: This method allows you to set the window handles as a string.
`set<string> set= driver.get.windowhandles()`
- `switch to`: This method aids in the switching of windows.
- `action`: This method aids in the execution of specific window actions.

Let us consider an example code to understand better. We will open the website of InterviewBit and then click on all the links available on the web page. Then, we will switch from the parent window to multiple different child windows and then switch back to the parent window at last.

```
package SeleniumPackage;
import java.util.Iterator;
import java.util.Set;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class WindowHandle_Demo {
    public static void main(String[] args) throws Exception {

        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\user\\Downloads\\seleni
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        // Loading the website
        driver.get("http://www.interviewbit.com/");
        String parent=driver.getWindowHandle(); // storing the parent window name as a str
        List<WebElement> links = driver.findElements(By.tagName("a")); // storing the list
        Iterator<WebElement> it = links.iterator();
        // Iterating over the list elements one by one and clicking all the links to open
        while(it.hasNext()){
            it.next().click();
        }
        Set<String> s = driver.getWindowHandles(); // Storing the list of all the child wi
        Iterator<String> I1= s.iterator();
        // Iterating over the list of child windows
        while(I1.hasNext())
        {
            String child_window=I1.next();
            if(!parent.equals(child_window))
            {
                driver.switchTo().window(child_window);
                System.out.println(driver.switchTo().window(child_window).getTitle());
                driver.close();
            }
        }
        //switch to the parent window
        driver.switchTo().window(parent);
    }
}
```

In the above code, we open the landing page of interviewbit and then find all the elements having the anchor tag and click them to open multiple child windows. Then, we iterate over each of the child windows and print them as a string. Finally, having traversed over the entire list, we break from the loop and switch back to the parent window.

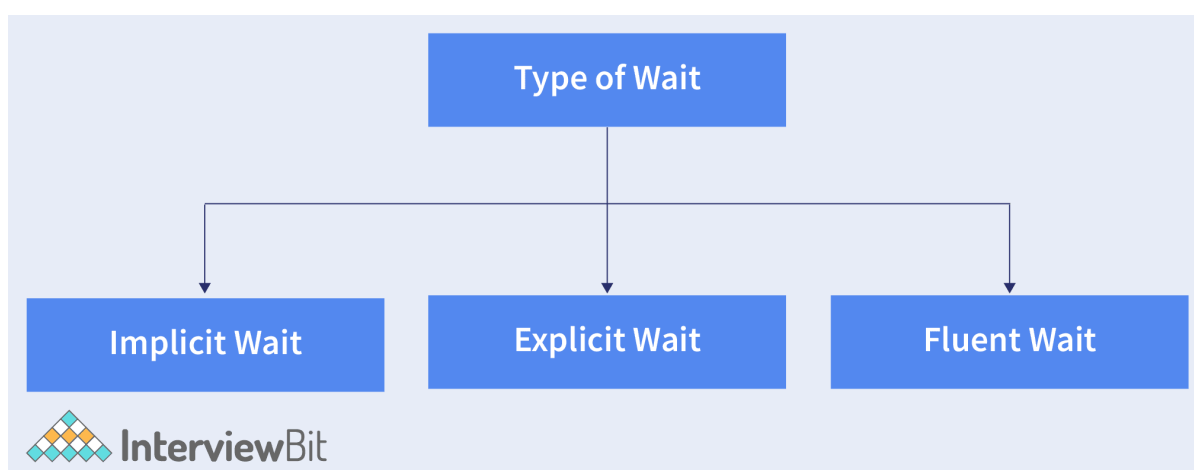
Selenium WebDriver Interview Questions

48. How to create an Object Repository in your project?

There is an Object Repository notion in QTP (**Quick Test Professional**). By default, when a user records a test, the objects and their properties are saved in an Object Repository. This Object Repository is used by QTP to playback scripts. There is no default Object Repository concept in Selenium. This isn't to say that Selenium doesn't have an Object Repository. Even if there isn't one by default, we could make our own.

Objects are referred to as locators in Selenium (such as ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, XPath, and CSS). A collection of objects is referred to as an object repository. Placing all the locators in a separate file is one technique to construct an Object Repository (i.e., properties file). The ideal method, however, is to use Page Object Model. Each web page is represented as a class in the Page Object Model Design Pattern. A class contains all of the items associated with a specific page of a web application.

49. What are the different types of waits that WebDriver supports?



1. Implicit Wait: Implicit wait instructs Selenium to wait a specified amount of time before throwing a "No such element" exception (One of the WebDriver Exceptions is NoSuchElementException, which occurs when the locators indicated in the Selenium Program code is unable to locate the web element on the web page).

Before throwing an exception, the Selenium WebDriver is told to wait for a particular amount of time. WebDriver will wait for the element after this time has been set before throwing an exception. Implicit Wait is activated and remains active for the duration of the browser's open state. Its default setting is 0, and the following protocol must be used to define the specific wait duration.

Its Syntax is as follows:

```
driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);
```

2. Explicit Wait: Explicit wait tells the WebDriver to wait for specific conditions before throwing an "ElementNotVisibleException" exception. The Explicit Wait command tells the WebDriver to wait until a certain condition occurs before continuing to execute the code. Setting Explicit Wait is crucial in circumstances when certain items take longer to load than others.

If an implicit wait command is specified, the browser will wait the same amount of time before loading each web element. This adds to the time it takes to run the test script. Explicit waiting is smarter, but it can only be used for specific parts. It is, nonetheless preferable to implicit wait since it allows the programme to stop for dynamically loaded Ajax items.

Its Syntax is as follows:

```
WebDriverWait wait = new WebDriverWait(WebDriver Reference, TimeOut);
```

3. Fluent Wait: It is used to inform the WebDriver how long to wait for a condition and how often we want to check it before throwing an "ElementNotVisibleException" exception. In Selenium, Fluent Wait refers to the maximum amount of time that a Selenium WebDriver will wait for a condition (web element) to become visible.

It also specifies how often WebDriver will check for the presence of the condition before throwing the "ElementNotVisibleException." To put it another way, Fluent Wait searches for a web element at regular intervals until it timeouts or the object is found. When engaging with site items that take longer to load, Fluent Wait instructions come in handy. This is a common occurrence with Ajax apps. It is possible to establish a default polling period while using Fluent Wait. During the polling time, the user can configure the wait to ignore any exceptions. Because they don't wait out the complete duration defined in the code, fluent waits are also known as Smart Waits.

Its Syntax is as follows:

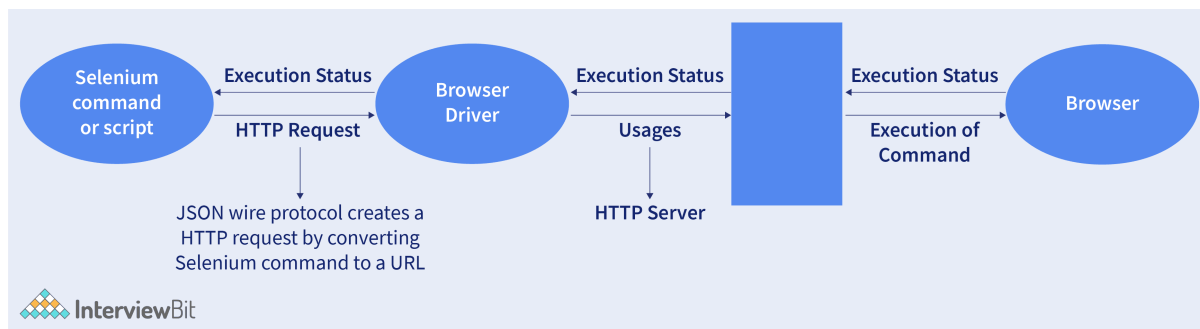
```
Wait wait = new FluentWait(WebDriver reference).withTimeout(timeout, SECONDS).pollingE\
```

50. Mention the several types of navigation commands that can be used?

The several types of navigation commands that can be used are as follows:

- `navigate().to()` - It is used for going to the specified URL.
- `driver.navigate().refresh()` - The current page is refreshed using the `driver.navigate().refresh()` command.
- `driver.navigate().forward()` - This command does the same function as clicking the browser's Forward Button. Nothing is accepted or returned by it.
- `driver.navigate().back()` - This command does the same function as clicking the browser's Back Button. Nothing is accepted or returned by it.

51. How does a Selenium WebDriver interact with the browser?



On a high level, the Selenium webdriver communicates with the browser and does not transform commands into JavaScript. Our Java or Python code will be transmitted as an API get and post request in the JSON wire protocol. The browser webdriver interacts with the real browser as an HTTP Request, as mentioned in the previous answer. To receive HTTP requests, each Browser Driver utilizes an HTTP server. When the URL reaches the Browser Driver, it will send the request via HTTP to the real browser. The commands in your Selenium script will be executed on the browser after this is completed.

If the request is a POST request, the browser will perform an action. If the request is a GET request, the browser will generate the corresponding response. The request will then be delivered to the browser driver through HTTP, and the browser driver will send it to the user interface via JSON Wire Protocol. [Learn More](#).

- The JSON wire protocol converts test commands into HTTP requests.
- Every browser has its own driver that initializes the server before executing any test cases.
- The browser's driver then begins to receive the request.

52. What are Selenium WebDriver Listeners?

Selenium WebDriver Listeners, as the name implies, "listen" to any event that the Selenium code specifies. Listeners are useful when you want to know what happens before you click any element, before and after you navigate to an element, or when an exception is thrown and the test fails. Listeners can be used in Selenium Automation Testing to log the order of activities and to capture a screenshot whenever an Exception is thrown. This makes debugging easier in the later stages of Test Execution. Some examples of Listeners are Web Driver Event Listeners and TestNG.

53. What is the implementation of WebDriver Listeners?

The Webdriver Event Listeners can be implemented in one of two ways:

- **WebDriverEventListener** is an interface with several built-in methods for tracking Webdriver events. It necessitates the implementation of ALL of the methods described in the Interface.
- The **AbstractWebDriverEventListener** class gives us the ability to implement only the methods that we're interested in.

54. In Selenium WebDriver, how do you handle Ajax calls?

When using Selenium WebDriver, one of the most prevalent challenges is handling AJAX calls. We would have no way of knowing when the AJAX call would complete, and the page would be refreshed. In this tutorial, we'll look at how to use Selenium to handle AJAX calls. AJAX (Asynchronous JavaScript and XML) is an acronym for Asynchronous JavaScript and XML. AJAX allows a web page to obtain little quantities of data from the server without having to completely reload the page. Without reloading the page, AJAX sends HTTP requests from the client to the server and then processes the server's answer. Wait commands may not work with AJAX controls. It's only that the page itself is not going to refresh.

The essential information may show on the web page without refreshing the browser when you click on a submit button. It may load in a fraction of a second, or it may take longer. We have no control over how long it takes for pages to load. In Selenium, the easiest way to deal with circumstances like this is to employ dynamic waits (i.e., `WebDriverWait` in combination with `ExpectedCondition`)

The following are some of the approaches that are available:

1. `titleIs()` – The anticipated condition looks for a specific title on a page.

```
wait.until(ExpectedConditions.titleIs("Big Sale of the Year"));
```

2. `elementToBeClickable()` – The desired condition requires that an element be clickable, which means that it must be present/displayed/visible on the screen and enabled.

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

3. `alertIsPresent()` – The expected condition anticipates the appearance of an alert box.

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

4. `textToBePresentInElement()` – The anticipated condition looks for a string pattern in an element.

```
wait.until(ExpectedConditions.textToBePresentInElement(By.id("text"),"text to be found"
```

55. Which implementation of WebDriver promises to be the fastest?

HTMLUnitDriver is the quickest WebDriver implementation because the HTMLUnitDriver does not run tests in the browser, this is the case. When compared to running the scripts without a browser, starting a browser and performing test cases took longer. For test case execution, HTMLUnitDriver used a simple HTTP request-response method.

56. At a bare minimum, how many parameters do selenium commands have?

The following four parameters need to be passed in Selenium:

1. Host: This is the parameter that binds Selenium to a particular IP address. Because we usually perform Selenium tests on our local system, the value will be 'localhost.' Instead of localhost, you can specify an IP address.

The Syntax is as follows:

```
java -jar <selenium server standalone jar name> -host <Your IP Address>
```

2. Port number: TCP/IP port for connecting Selenium tests to the Selenium Grid Hub. 4444 is the default port hub.

The Syntax is as follows:

```
java -jar <selenium server standalone jar name> -role hub -port 4444
```

Assure this port isn't being used by any other software on your machine. An exception like Exception in thread "main" java.net may occur. Selenium is already running on port 4444. BindException: Selenium is already running on port 4444. Alternatively, some other service is available.

If this happens, you may either kill the other process using port 4444 or tell Selenium-Grid to use a new port for its hub. If you want to change the hub's port, the -port option can be used.

3. Browser: For the execution of the selenium scripts I required a browser to be passed.

4. Url: The URL of the application needs to be passed.

57. As seen below, we establish a WebDriver reference variable called 'driver.' What exactly is the purpose of proceeding in this manner?

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

instead of creating

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

We may use the same driver variable to work with any browser we want, such as IEDriver, SafariDriver, and so on if we construct a reference variable of type WebDriver.

58. What kinds of Selenium WebDriver exceptions have you run into?

In the following project the exceptions we've run into are as follows:

- **Element Not Visible Exception:** This error is produced when you try to discover a certain element on a webpage that is not currently accessible, despite the fact that it exists in the DOM. Also, if you're trying to find an element with an XPath that connects two or more elements, it can be difficult.
- **Stale Element Reference Exception:** This is thrown in one of two scenarios, the first of which is more prevalent:
 - The element has been completely removed.
 - The element has been detached from the DOM.

On the elements, with which we are interacting, destruction and then restoration, we get a stale element reference exception. When this happens, the element's DOM reference becomes invalid. Because of this, we are unable to obtain the element's reference.

The following are some more common exceptions:

- **WebDriverException**
- **TimeoutException**
- **NoAlertPresentException**
- **NoSuchWindowException**
- **IllegalStateException**
- **NoSuchElementException.**

59. What is the best way to deal with StaleElementReferenceException?

Before we look at how to manage a StaleElementReferenceException using the Page Object Model, let's have a look at how to handle a StaleElementReferenceException.

First, let's define Stale Element Reference Exception. Stale refers to something that is old, deteriorated, and no longer fresh. An old or no longer available element is referred to as a stale element. Assume there is a WebElement in WebDriver that is discovered on a web page. The WebElement becomes stale when the DOM changes. The StaleElementReferenceException is thrown when we try to interact with a stale element.

60. In a Selenium script, what happens if you use both implicit and explicit wait?

According to the official Selenium manual, mixing Implicit and Explicit Waits is not recommended. Combining the two can result in unpredictable wait times. Only one time in the code is implicit wait specified. Throughout the driver object instance, it will remain the same.

Explicit wait is used in the code whenever it is required. At the time of execution, this wait will be called. It's a conditional waiting period. Wherever explicit wait is applied, it will supersede the implicit wait. As a result, **Explicit Wait takes precedence over Implicit Wait.**

61. In a Selenium Script, what happens if you use both Thread.Sleep and WebDriver Waits?

The `Thread.sleep()` method allows you to suspend the execution for a specified amount of time in milliseconds. If we use WebDriver waits in conjunction with the `Thread.sleep()` method, the webdriver will pause the execution for the provided amount of time in the parameter of the `Thread.sleep()` function before proceeding to the next wait. If we combine both waits, the test execution time will increase.

62. In Selenium WebDriver, how do you take a screenshot?

During the execution of the test scripts, test cases may fail. We just capture a screenshot and save it in a result repository while manually executing the test cases. Selenium WebDriver can be used to accomplish the same thing. Some of the instances in which we might need to use Selenium WebDriver to capture a screenshot are as follows:

- Problems with applications
- Assertion Defect
- Finding Web Elements on the web page is difficult.
- Take a break to look for Webelements on the page.

TakesScreenshot is a Selenium interface that has a function `getScreenShotAs` that may be used to take a screenshot of the programme under test. When capturing screenshots with Selenium 3, we may run into a few difficulties. We utilize the `aShot()` function to get around this.

63. How do I use Selenium WebDriver to enter text into a text box?

Using the method `sendKeys()`

```
WebDriver driver = new FirefoxDriver();
driver.get("https://www.youtube.com");
driver.findElement(By.xpath("xpath")).sendKeys("Interview Bit");
```

64. How can I type text into the text box without using the `sendKeys()` function?

We can use the following piece of code to type text into the text box without using the `sendKeys()` function:

```
// To initialize js object
JavascriptExecutor JS = (JavascriptExecutor)webdriver;
// To enter username
JS.executeScript("document.getElementById('User').value='InterviewBit.com'");
// To enter password
JS.executeScript("document.getElementById('Pass').value='tester value'");
```

65. How can I use Selenium WebDriver to clear the text in a text box?

The above task can be done using the `clear()` function as shown below:

```
WebDriver driver = new FirefoxDriver();
driver.get("https://www.youtube.com");
driver.findElement(By.xpath("xpath_of_element1")).sendKeys("Interview Bit");
driver.findElement(By.xpath("xpath_of_element1")).clear();
```

66. What is the best way to acquire the textual matter of a web element?

The best way to acquire the textual matter of a web element is by employing the `getText()` method as shown below:

```
package interviewBit;
// package name
import org.openqa.selenium.By;
import org.testng.annotations.Test;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.WebDriver;
// importing the necessary libraries
// Test class
public class Test {
    @Test
    public void testmethod(){
        // sets the property as required
        System.setProperty("webdriver.chrome.driver", "C:\\Selenium Environment\\Drivers\\chromedriver.exe");
        //Creates a new Chrome Web Driver
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.youtube.com");
        String textAvailable=driver.findElement(By.xpath("//*[@id='gbw']/div/div/div[1]/div[1]/div[1]")).getText();
        System.out.println("Textual Matter which is Present is :"+ textAvailable);
    }
}
```

67. How do I retrieve the value of an attribute in Selenium WebDriver?

Using the `getAttribute(value)` method. It returns the value of the parameterized attribute.

HTML:

```
<input name="nameSeleniumWebDriver" value="valueSeleniumWebDriver">Interview Bit</input>
```

Selenium Program:

```
String attributeValue = driver.findElement(By.name("nameSeleniumWebDriver")).getAttribute("value");
System.out.println("Available attribute value is :"+attributeValue);
```

Output:

```
valueSeleniumWebDriver
```


68. How can I use Selenium WebDriver for clicking on a hyperlink?

In Selenium, we use the `click()` method for clicking on a hyperlink.

```
driver.findElement(By.linkText("Interview Bit Website")).click();
```

69. How do I use Selenium WebDriver for submitting a form?

For submitting a form in Selenium WebDriver, we use the "submit" method on the element.

```
driver.findElement(By.id("form")).submit();
```

70. In Selenium WebDriver, how do I push the ENTER key on a text box?

To utilize Selenium WebDriver to hit the ENTER key, we must use Selenium Enum Keys with the constant ENTER.

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

71. How can I use WebDriver to mouse hover over a web element?

We can use the Actions class to mouse hover over a web element as shown below in the code snippet:

```
WebElement ele = driver.findElement(By.xpath("xpath"));  
//Create object 'action' of an Actions class  
Actions action = new Actions(driver);  
//Mouseover on an element  
action.moveToElement(ele).perform();
```

72. How does Selenium WebDriver handle hidden elements?

```
(JavascriptExecutor(driver)).executeScript("document.getElementsByClassName(ElementLoca
```

73. In Selenium WebDriver, how do you use the Recovery Scenario?

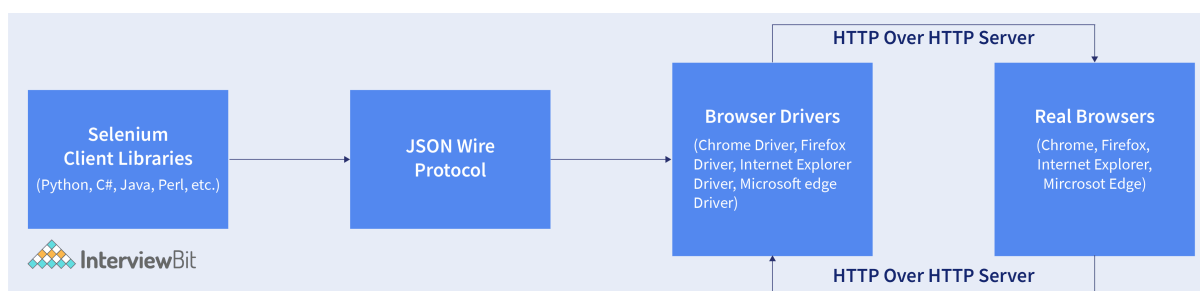
Within Selenium WebDriver Java tests, by using "Try Catch Block."

```
try {  
    driver.get("www.interviewbit.com");  
}catch(Exception e){  
    System.out.println(e.getMessage());  
}
```

74. What is the Selenium WebDriver Architecture?

Selenium WebDriver is a web framework for performing cross-browser testing (Cross-browser testing is comparing and assessing your website's behaviour in various browser contexts. It ensures that your website provides the best possible user experience regardless of the browser used to access it). This program is used to test web-based applications to ensure that they work as expected. Selenium WebDriver allows us to write test scripts in the programming language of our choice. As previously stated, it is an improvement to Selenium RC in that it overcomes a few shortcomings. Although Selenium WebDriver is unable to handle window components, this limitation can be bypassed by using tools such as Sikuli, Auto IT, and others.

The [Selenium WebDriver Architecture](#) consists of:



- Selenium Client library.
- JSON wire protocol over HTTP.
- Browser Drivers.
- Browsers.

Selenium Tricky Interview Questions

75. Why is it important to use TestNG when working with Selenium RC?

TestNG is a testing framework that provides more powerful and flexible test configuration and execution than JUnit, the default testing framework used with Selenium RC. TestNG allows for the annotation of test methods, support for data-driven testing, and the ability to run tests in parallel, which can lead to faster test execution and better test organization. Additionally, TestNG provides more advanced reporting and logging capabilities than JUnit. Using TestNG with Selenium RC can help improve the efficiency and effectiveness of automating web application testing.

76. Can Selenium be used to test responsive web design?

Yes, Selenium can be used to test responsive web design. Selenium is a web automation tool that allows for the automation of browser interactions, and as such, it can be used to test the responsiveness of a web page by simulating interactions on different screen sizes and resolutions. This can include testing how the page layout adjusts to different viewport sizes, testing how the page elements are displayed on different devices, and testing how the page behaves when a user switches between landscape and portrait orientations.

One way to test responsive web design with Selenium is by using the Selenium WebDriver API to set the browser's viewport size and then test the page's layout and functionality. Another way is to use a third-party library such as Selenium-Responsive-Testing which allows you to test how the page behaves on different devices and screen sizes.

It is worth noting that Selenium alone is not enough for responsive web design testing, it should be combined with other methods such as manual testing, visual testing, and using responsive design testing tools to get complete coverage.

77. What API should be used to test databases when using Selenium WebDriver for database testing?

Selenium WebDriver is a tool for automating web browsers, it's not designed for database testing. To test databases, you need a database testing tool that can connect to the database and perform various operations like running SQL queries and asserting the results. Some examples of database testing tools are DBUnit, jOOQ, and JDBC. These tools provide an API that can be used to connect to the database, perform operations and assert the results.

78. What are assertions in Selenium?

In the context of Selenium, assertions are used to check if the actual outcome of a test matches the expected outcome. Assertions are used to validate that a certain condition is true and if it's not true, the test execution will be stopped and an assertion error will be thrown. Assertions can be used to verify that a specific element is present on the page, to check the text of an element, to check the value of an input field, and many other things. Selenium WebDriver provides assertion methods through the `org.junit.Assert` and `org.testng.Assert` classes for JUnit and TestNG respectively.

79. What is the Silk Test Tool?

Silk Testing (formerly known as Segue Silk Test) is a commercial, functional, and regression testing tool used for automating the testing of software applications. It is developed and maintained by Micro Focus and it is used for automating GUI-based tests for a variety of applications, including web, mobile, and desktop applications. The tool can be used to test applications on different platforms such as Windows, iOS, and Android. Silk Testing uses a proprietary scripting language called 4Test, which allows users to automate the testing process by writing test scripts that simulate user interactions with the application under test. The tool also has a visual text editor that allows users to record and edit test scripts using a GUI.

80. What is the purpose of the testing.xml file ?

The testing.xml file, also known as testng.xml is an XML configuration file used in the TestNG framework. It is used to configure and execute test suites and test cases. It provides a way to define and organize test methods, classes, and packages into test suites. It also allows setting up test execution order, parallel execution, test data, test listeners, test reporting, and more. With the testing.xml file, users can specify which test methods or classes should be included or excluded in a test run, what test listeners should be used, and what test configuration should be used, among other things. It is used to organize and execute test methods, classes, and packages in a logical and organized way.

81. What are the areas where Selenium can improve its features?

Selenium has a wide range of capabilities, but some areas where it could see the benefit in extending its features include:

- **Mobile testing:** Selenium has limited support for mobile testing. It can automate web applications on mobile browsers, but it does not support automating native mobile applications.
- **Visual testing:** Selenium is mainly used for functional testing, but it does not have built-in support for visual testing. Visual testing compares the look and feel of the application to a pre-approved design and helps in identifying issues related to layout, alignment, and font.
- **Cross-browser testing:** Selenium supports a wide range of browsers, but there are still some browsers that are not fully supported or have limited support.
- **Test reports and analysis:** Selenium's test reporting capabilities are basic and it does not provide advanced test analytics features like test coverage, flaky test detection, and root cause analysis.
- **Performance testing:** Selenium is mainly used for functional testing, but it does not have built-in support for performance testing. Performance testing can help to measure the responsiveness, stability, and scalability of the application under test.
- **Continuous Integration:** Selenium does not provide out-of-the-box support for continuous integration (CI) and continuous delivery (CD) pipeline, it needs to be integrated with other tools to automate the test execution in CI/CD pipeline.

It's worth noting that Selenium is widely used and has a large community of developers who are constantly working to improve and extend its capabilities. There are also many other test automation tools that can be used in combination with Selenium to address these limitations.

82. Can you explain what Page Factory is?

Page Factory is a design pattern used to create an object repository for web elements in Selenium WebDriver. It's an extension of the Page Object Model design pattern, which is used to create a centralized repository for web elements. It uses annotations to identify and initialize web elements at runtime and make the code more readable and maintainable by separating the test code from the technical details of locating web elements. In order to use Page Factory, you need to create a class for each web page and define web elements in that class using the `@FindBy` annotation it also contains methods for interacting with the web elements.

83. What is the Actions class?

The Actions class is a part of the Selenium Python bindings for automating web browsers. It allows you to perform actions on web elements, such as moving the mouse over an element, clicking on it, and sending keys to it. The Actions class is typically used in conjunction with the WebDriver class, which is the main class for interacting with a web browser.

84. What are the steps for troubleshooting tests using Selenium IDE?

Troubleshooting tests using Selenium IDE can be a multi-step process, but here are some general steps that you can follow:

- Verify that the test is correctly recorded: Make sure that the test is recorded correctly and that all the elements and commands are in the correct order.
- Check the test for syntax errors: Selenium IDE will alert you if there are any syntax errors in the test, so make sure to check for and fix any errors.
- Check the test for compatibility issues: Make sure that the test is compatible with the browser and version that you are using to run the test.
- Check the test for missing elements: Make sure that all the elements used in the test are present on the web page.
- Run the test in debug mode: Selenium IDE provides a debug mode that allows you to step through the test and check the values of variables and the state of the web page at each step.
- Check the browser's developer tools: The browser's developer tools can be used to inspect the web page and check for any issues.
- Check the log messages: Selenium IDE provides a log panel where you can view log messages generated by the test. This can be useful for troubleshooting issues.
- Check the Selenium documentation: The Selenium documentation provides information on the different commands and options available in Selenium IDE, which can be helpful for troubleshooting.
- Finally, you can also seek help from the Selenium community and forums, where you can ask questions and get help from other Selenium users.

85. What steps can be taken to resolve an issue where a Selenium script only works properly on Google Chrome but not Internet Explorer?

Here are some possible steps that can be taken to resolve an issue where a Selenium script only works properly on Google Chrome but not Internet Explorer:

1. Verify that the correct version of Internet Explorer is being used: Selenium supports different versions of Internet Explorer, so make sure that you are using a version that is compatible with your Selenium script.
2. Check the Internet Explorer settings: Make sure that Internet Explorer's security settings are configured correctly and that the browser is not running in compatibility mode.
3. Check the Selenium WebDriver version: Make sure that you are using the latest version of the Selenium WebDriver for Internet Explorer.
4. Check the browser's developer tools: Use the browser's developer tools to inspect the web page and check for any issues.
5. Check the log messages: Selenium provides log messages that can be helpful for troubleshooting issues.
6. Check the Internet Explorer documentation: The Internet Explorer documentation provides information on the different options and settings available in Internet Explorer, which can be helpful for troubleshooting.
7. Use a different browser: If the above steps do not resolve the issue, you may want to consider using a different browser that is supported by Selenium, like Firefox or Microsoft Edge.
8. You can also try to seek help from the Selenium community and forums, where you can ask questions and get help from other Selenium users.

86. Is it possible to open pop-up windows with Selenium?

Yes, it is possible to open pop-up windows with Selenium. Selenium WebDriver provides methods to interact with browser windows, so you can use Selenium to open new windows, switch between windows, and close windows.

You can open a new window using the `window_handles` method, and switch to the new window using the `switch_to.window()` method. Once you have switched to the new window, you can interact with the elements on the pop-up window just like you would interact with elements on the main window.

You can also use the `Alert` class of Selenium WebDriver to handle JavaScript alerts and confirmations.

It's worth noting that, while some browsers may block pop-ups by default, you need to configure the browser settings to allow pop-ups for the testing website.

It's also important to use explicit waits to handle the dynamic nature of the web pages and pop-ups, as the time taken for a pop-up window to load may vary depending on the browser, internet connection, and other factors.

87. Can selenium be used to launch web browsers?

Yes, Selenium provides good support to launch web browsers like Google Chrome, Mozilla Firefox, Internet Explorer, etc.

The following commands can be used to launch web browsers using Selenium:

- `WebDriver driver = new FirefoxDriver();`
- `WebDriver driver = new ChromeDriver();`
- `WebDriver driver = new InternetExplorerDriver();`

88. Is it possible to use only perform() without build()?

Yes, it is possible to use the perform() method without calling the build() method. The build() method is used to construct and return an instance of the ActionChains class, which is a container for a series of actions. However, if you don't call build(), the actions will be performed immediately when you call perform().

```
actions.move_to_element(element).click().perform()
```

The above line of code is an example of using perform() without calling build(). The move_to_element(), click() methods are chained together and the perform() method is used to execute all the chained actions.

However, it is important to note that, perform() method is used to execute all the chained actions only after calling build() when you are chaining multiple actions together and want to execute them together.

89. What is StaleElementReferenceException? When does this occur? And how to overcome such exceptions?

StaleElementReferenceException is an exception that is thrown in Selenium WebDriver when a web element that was previously found on a web page is no longer available or has been deleted from the DOM (Document Object Model). This can happen if the web page has been refreshed, the element has been removed or replaced, or the element's parent element has been removed or replaced.

This exception occurs when the element is no longer attached to the DOM, which means that the element can no longer be interacted with using the WebDriver.

There are a few ways to overcome this exception:

- **Re-find the element:** You can re-find the element by using the WebDriver's `find_element` or `find_elements` methods again.

```
try:
    element.click()
except StaleElementReferenceException:
    element = driver.find_element_by_id('element_id')
    element.click()
```

- **Wait for the element to be available:** You can use explicit waits such as `WebDriverWait` and `ExpectedConditions` to wait for the element to be available before interacting with it.

```
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait

try:
    element.click()
except StaleElementReferenceException:
    wait = WebDriverWait(driver, 10)
    element = wait.until(EC.presence_of_element_located((By.ID, 'element_id')))
    element.click()
```

- **Refresh the page:** You can refresh the page using the `refresh()` method of the WebDriver instance and then re-find the element.

```
try:
    element.click()
except StaleElementReferenceException:
    driver.refresh()
    element = driver.find_element_by_id('element_id')
    element.click()
```

It is important to note that this exception is a run-time exception, so it's best to catch it and handle it in the code, rather than letting the script crash.

90. What are test design techniques? Explain BVA and ECP with some examples.

Test design techniques are the methods or strategies used to design and create test cases for a software application. Some common test design techniques include:

1. **Black Box Testing:** Testing the functionality of the software without looking into the internal structure or code.
2. **White Box Testing:** Testing the internal structure or code of the software.
3. **Grey Box Testing:** Combining both Black Box and White Box testing methods.
4. **Boundary Value Analysis (BVA):** This technique is used to test the software's behaviour at its input and output boundaries. It involves testing the software with input values that are just above and just below the valid range, as well as testing the software with the maximum and minimum valid input values.
5. **Equivalence Class Partitioning (ECP):** This technique is used to divide the input domain of the software into a finite number of classes or partitions, where each class represents a group of input values that are expected to behave in the same way. The goal of ECP is to test the software with a representative sample of input values from each partition.

Examples:

BVA:

1. In a login form, test the minimum and maximum length of the password field.
2. For a form field that accepts a date, testing the software with a date that is just before and just after the valid range of dates.

ECP:

1. In a form field that accepts a phone number, divide the input domain into three classes: valid phone numbers, phone numbers that are too short, and phone numbers that are too long.
2. In a form field that accepts a price, divide the input domain into two classes: valid prices and invalid prices (e.g., negative prices).

It is important to note that these techniques are helpful in creating a comprehensive set of test cases that can help ensure that the software is functioning correctly.

However, it's not enough to just use one technique, a combination of different techniques are used to test the software in different ways and identify all the possible defects.

Recommended Resources

- [RPA Interview Questions and Answers](#)
- [Java Interview Questions and Answers](#)
- [Automation Testing Interview Questions and Answers](#)
- [Selenium Interview Questions for 5years Experience](#)
- [Cucumber Interview Questions and Answers](#)
- [Appium Interview Questions and Answers](#)
- [Database Testing Interview Questions](#)
- [Technical Interview Guide](#)
- [Principles of Software Testing](#)
- [Difference Between Alpha and Beta Testing](#)

Links to More Interview Questions

[C Interview Questions](#)

[Php Interview Questions](#)

[C Sharp Interview Questions](#)

[Web Api Interview Questions](#)

[Hibernate Interview Questions](#)

[Node Js Interview Questions](#)

[Cpp Interview Questions](#)

[Oops Interview Questions](#)

[Devops Interview Questions](#)

[Machine Learning Interview Questions](#)

[Docker Interview Questions](#)

[Mysql Interview Questions](#)

[Css Interview Questions](#)

[Laravel Interview Questions](#)

[Asp Net Interview Questions](#)

[Django Interview Questions](#)

[Dot Net Interview Questions](#)

[Kubernetes Interview Questions](#)

[Operating System Interview Questions](#)

[React Native Interview Questions](#)

[Aws Interview Questions](#)

[Git Interview Questions](#)

[Java 8 Interview Questions](#)

[Mongodb Interview Questions](#)

[Dbms Interview Questions](#)

[Spring Boot Interview Questions](#)

[Power Bi Interview Questions](#)

[Pl Sql Interview Questions](#)

[Tableau Interview Questions](#)

[Linux Interview Questions](#)

[Ansible Interview Questions](#)

[Java Interview Questions](#)

[Jenkins Interview Questions](#)