InterviewBit

Selenium WebDriver Interview Questions

To view the live version of the page, click here.

© Copyright by Interviewbit

Contents

Selenium WebDriver Interview Questions for Freshers

1. What is Selenium WebDriver?
2. Is Selenium WebDriver a library?
3. Is Selenium WebDriver an interface or a class?
4. What are the different types of WebDriver Application Programming Interfaces in Selenium?
5. What programming languages does Selenium WebDiver support?
6. What open-source frameworks does Selenium WebDriver support?
7. What is WebDriver's super interface?
8. Explain the following line of code.
9. Is it necessary to use Selenium Server to run Selenium WebDriver scripts?
10. What will happen if I execute this command? driver.get("www.interviewbit.com") ;
11. What is an alternative option to driver.get() method to open an URL in Selenium Web Driver?
12. What is the difference between driver.get() and driver.navigate.to("url")?
13. In Selenium WebDriver, what is the difference between driver.getWindowHandle() and driver.getWindowHandles()?
14. What are the differences between the methods driver.close() and driver.quit()?
15. What is the difference between driver.findElement() and driver.findElements() commands?
16. What are some cases that Selenium WebDriver cannot automate?
17. In Selenium WebDriver, what is an Object Repository?

Selenium WebDriver Interview Questions for Experienced

InterviewBit

© Copyright by Interviewbit

Selenium WebDriver Interview Questions for Experienced

(.....Continued)

19. What are the different types of waits that WebDriver supports?
20. Mention the several types of navigation commands that can be used?
21. How does a Selenium WebDriver interact with the browser?
22. What are the components of Selenium Suite?
23. What are Selenium WebDriver Listeners?
24. What is the implementation of WebDriver Listeners?
25. What are the drawbacks of using Selenium for testing?
26. In Selenium WebDriver, how do you handle Ajax calls?
27. Which implementation of WebDriver promises to be the fastest?
28. At a bare minimum, how many parameters do selenium commands have?
29. As seen below, we establish a WebDriver reference variable called 'driver.' What exactly is the purpose of proceeding in this manner?
30. What kinds of Selenium WebDriver exceptions have you run into?
31. What is the best way to deal with StaleElementReferenceException?
32. In a Selenium script, what happens if you use both implicit and explicit wait?
33. In a Selenium Script, what happens if you use both Thread.Sleep and WebDriver Waits?
34. In Selenium WebDriver, how do you take a screenshot?
35. How do I use Selenium WebDriver to enter text into a text box?
36. How can I type text into the text box without using the sendKeys() function?
37. How can I use Selenium WebDriver to clear the text in a text box?
38. What is the best way to acquire the textual matter of a web element?

InterviewBit

© Copyright by Interviewbit

Selenium WebDriver Interview Questions for Experienced

(.....Continued)

39. How do I retrieve the value of an attribute in Selenium WebDriver?
40. How can I use Selenium WebDriver for clicking on a hyperlink?
41. How do I use Selenium WebDriver for submitting a form?
42. In Selenium WebDriver, how do I push the ENTER key on a text box?
43. How can I use WebDriver to mouse hover over a web element?
44. How does Selenium WebDriver handle hidden elements?
45. In Selenium WebDriver, how do you use the Recovery Scenario?

InterviewBit

© Copyright by Interviewbit

Let's get Started

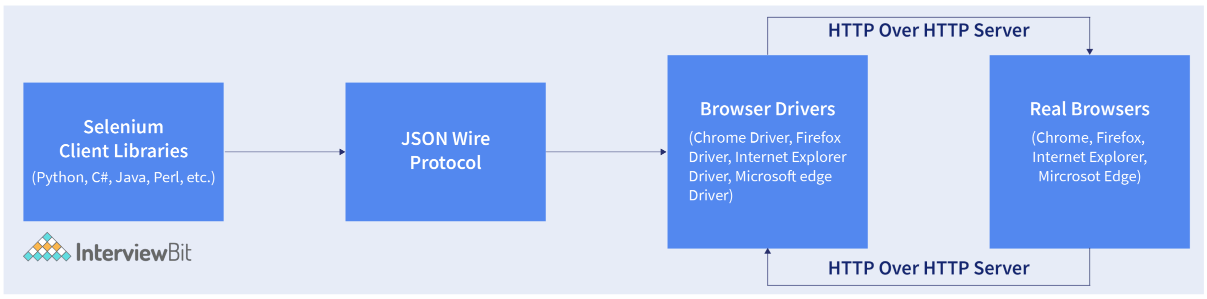
Introduction:

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.

InterviewBit

Let us have a look at the Selenium WebDriver Architecture:

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


Selenium Client Libraries
(Python, C#, Java, Perl, etc.)
JSON Wire Protocol
Browser Drivers
(Chrome Driver, Firefox Driver, Internet Explorer Driver, Microsoft edge Driver)
Real Browsers
(Chrome, Firefox, Internet Explorer, Mircrosot Edge)
HTTP Over HTTP Server
HTTP Over HTTP Server
InterviewBit

Client Libraries:

© Copyright by Interviewbit

Selenium supports a variety of libraries, including Ruby, Python, Java, and others, thanks to language bindings created by Selenium developers to ensure language compatibility. In order to use the browser driver in Python, we can use the Python Bindings. From Selenium's official website, you can download all of the supported language bindings of your choice.

JSON Wire Protocol (JSON Wire Protocol):

The acronym JSON stands for JavaScript Object Notation. It is an open standard that provides a web-based transport method for data transfer between clients and servers. It supports numerous data structures such as arrays and objects, making it easy to read and write JSON data.

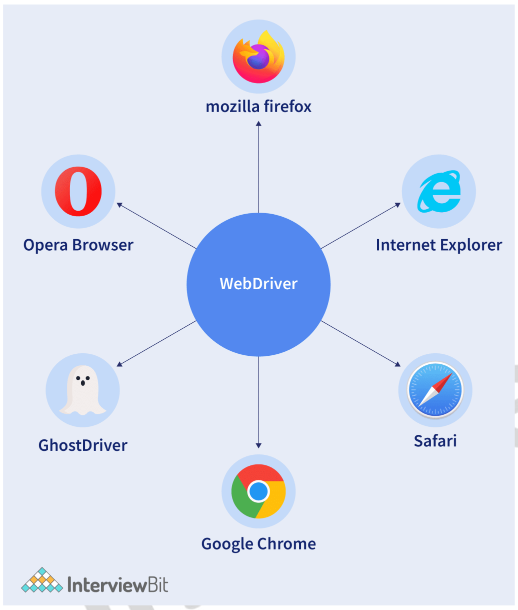
Drivers for Web Browsers:

Selenium provides drivers for each browser, and the browser driver communicates with the relevant browser by creating a secure connection, without revealing the core logic of browser capabilities. These browser drivers are also particular to the test case automation language, such as C#, Python, or Java. You can choose your preferred browser driver based on your linguistic requirements. On BrowserStack, for example, you can set up the Selenium WebDriver for Python. When using WebDriver to run a test script, the following tasks are performed in the background:

- For each Selenium Command, an HTTP request is generated and provided to the browser driver.
- The driver receives the HTTP request from an HTTP server.
- An algorithm determines all of the steps/instructions to be executed in the browser.
- After that, the HTTP server receives the execution status and delivers it back to the automation scripts.

Selenium supports a variety of browsers, including Chrome, Firefox, Safari, and Internet Explorer:

© Copyright by Interviewbit



Here in this article, we've covered all the important interview questions on selenium webdriver to help you ace your interviews.

Selenium WebDriver Interview Questions for Freshers

1. 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.

2. Is Selenium WebDriver a library?

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

3. 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.

4. 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.

5. What programming languages does Selenium WebDiver support?

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

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

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

© Copyright by Interviewbit

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.

InterviewBit

7. What is WebDriver's super interface?

SearchContext is the Super Interface of the WebDriver.

8. 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).

9. 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.

10. What will happen if I execute this command? driver.get(“www.interviewbit.com”) ;

© Copyright by Interviewbit

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.

11. 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.

12. 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.

13. 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.

14. 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()` .

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

InterviewBit

© Copyright by Interviewbit

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"));
```

16. 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.

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

© Copyright by Interviewbit

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 WebDriver Interview Questions for Experienced

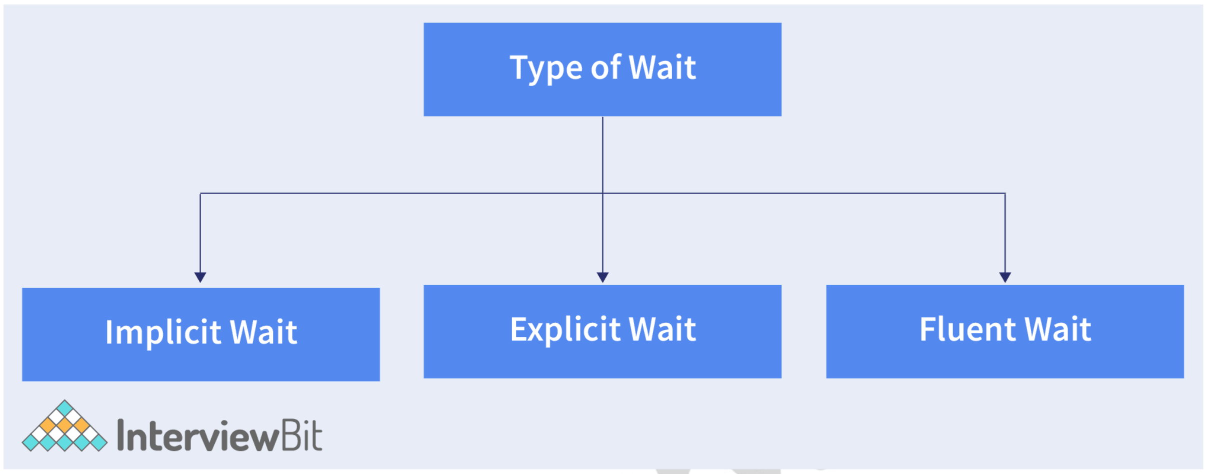
18. How to create an Object Repository in your project?

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.

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

© Copyright by Interviewbit



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 are 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. It's 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.

© Copyright by Interviewbit

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).pollingEv
```

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

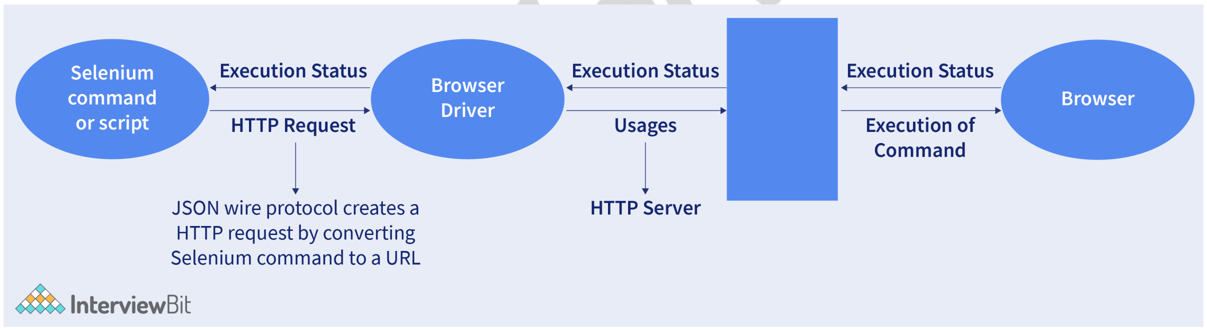
The several types of navigation commands that can be used are as follows:

© Copyright by Interviewbit

- `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.

InterviewBit

21. 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.

© Copyright by Interviewbit

- 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.

22. What are the components of Selenium Suite?

The components of Selenium Suite are as follows:

- **IDE Selenium:** It is a Firefox/Chrome add-on designed to make writing automation scripts go faster. It captures user activities in the web browser and saves them as a script that may be reused.
- **Remote Control for Selenium (RC):** RC is a server that lets users write application tests in a number of different programming languages. This server accepts the commands from the test script and sends them to the browser as Selenium core JavaScript commands. The browser then responds appropriately.
- **WebDriver for Selenium:** WebDriver is a programming interface that aids in the development and execution of test cases. It includes the ability to operate on web items. WebDriver, unlike RC, does not require a separate server and interacts directly with browser apps.
- **Grid of Selenium:** The grid was created to deliver commands to multiple machines at the same time. It enables the running of tests in parallel across multiple browsers and operating systems.

23. 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.

24. What is the implementation of WebDriver Listeners?

InterviewBit

© Copyright by Interviewbit

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.

25. What are the drawbacks of using Selenium for testing?

The drawbacks of using Selenium for testing are as follows:

- **Unavailability of dependable technical assistance**: Because Selenium is an open source technology, there is no dedicated tech support to help users with their problems.
- **Only tests web applications**: To test desktop and mobile applications, Selenium must be combined with third-party tools such as Appium and TestNG.
- **Image Testing Limitations**: Image testing has limited support.
- **No in-built reporting**: Selenium also lacks a built in reporting and test management feature, necessitating integration with tools such as TestNG or JUnit, among others, in order to allow test reporting and management. It's possible that you'll need to know how to program in one of the following languages: Selenium WebDriver assumes that the user has some programming experience.

26. 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.

© Copyright by Interviewbit

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'
```

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

© Copyright by Interviewbit

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.

28. 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 browser is passed.

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

© Copyright by Interviewbit

29. 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.

InterviewBit

30. 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 element's, 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:

© Copyright by Interviewbit

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

31. 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.

InterviewBit

32. 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.

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

© Copyright by Interviewbit

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.

34. 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.

35. 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");
```

36. 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'");
```

37. 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();
```

38. 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:

© Copyright by Interviewbit

```
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\\chrom
//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]/
  System.out.println("Textual Matter which is Present is :" + textAvailable);
}
}
```

39. 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")).getAttribu
System.out.println("Available attribute value is :"+attributeValue);
```

Output:

```
valueSeleniumWebDriver
```

© Copyright by Interviewbit

40. 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();
```

41. 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();
```

42. 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);
```

43. 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();
```

44. How does Selenium WebDriver handle hidden elements?

© Copyright by Interviewbit

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

45. 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());
}
```

Useful Interview Resources:

- [Automation Testing Interview Questions]
- [Software Testing Interview Questions]

© Copyright by Interviewbit

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

© Copyright by Interviewbit