Selenium Interview Questions & Answers

Question 1:

Which WebDriver command takes you forward by one page from the browser's history?

a. navigate().forward()
b. navigate_forward()
c. Navigate.forward
d. navigate.forward()
e. Navigate.forward()

a. navigate().forward()

Question 2:

Which of the following code can be used to create excel automation object using POI excel API?

a. File excelobj = new File ();
b. excelobj = new File ();
c. File excelobj = File ();
d. File excelobj == new File ();

a. File excelobj = new File(); can be used to create an excel file object using POI excel API.

Here's an example of how to create an excel object using POI:

```
import java.io.File;
import java.io.IOException;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class ExcelFileExample {
    public static void main(String[] args) {
        // Creating an excel file object
        File excelFile = new File("path/to/excel.xlsx");
```

```
        // Creating a workbook object
        XSSFWorkbook workbook = null;
        try {
            workbook = new XSSFWorkbook(excelFile);
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Close the workbook
        try {
            workbook.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In this example, the File class is used to create an object of the excel file, and the XSSFWorkbook class from Apache POI library is used to create a workbook object that represents the excel file. The workbook can then be used to read or write data to the excel file.

Question 3:

Which of the following command is a “OnEvent Handler”?

a. focus()

b. assertAlert()

c. fireEvent()

d. alert()

c. fireEvent() is an "OnEvent Handler" command in Selenium. The fireEvent() method is used to trigger JavaScript events such as onclick, onchange, etc. in the web page being tested. The method takes two arguments: the first argument is the

locator of the web element on which the event is to be triggered and the second argument is the name of the event to be triggered.

Example:

```
WebElement element = driver.findElement(By.id("elementId"));
JavascriptExecutor executor = (JavascriptExecutor) driver;
executor.executeScript("arguments[0].fireEvent('onclick');", element);
```

Note: The fireEvent() method is available in Selenium RC and is not supported in Selenium WebDriver. In Selenium WebDriver, you can use the click() method to simulate a click event.

Question 4:

Would the two statements given below retrieve the title of a web page?

(Note:click on Question to enlarge)

a. Yes
b. No

Question 5:

What is the similarity between WebDriver's close() and quit() methods?

a. Closes all opened web browser windows.
b. Closes the active web browser window.
c. Does not accept arguments.
d. None of these

b. Closes the active web browser window.

Both the close() and quit() methods in WebDriver are used to close a web browser window. However, the close() method only closes the active window, whereas the quit() method closes all windows associated with the current WebDriver session, effectively ending the session and releasing all associated resources.

Question 6:

Which of the following WebDriver is used for Headless browser testing?

a. HeadLessDriver
b. FireFoxDriver
c. InternetExplorerDriver
d. HtmlUnitDriver

d. HtmlUnitDriver

Headless browser testing is a type of testing where a browser is executed without a graphical user interface (GUI). HtmlUnitDriver is a type of WebDriver that can be used for headless browser testing. HtmlUnitDriver is a pure Java implementation of a headless browser, which means that it doesn't require any browser to be installed on the system. This makes it a popular choice for headless browser testing as it is lightweight and fast, and can run on various platforms without any compatibility issues.

Question 7:

In WebDriver, which of the following command would you find the best to enter values into a text box?

a. type()
b. sendKeys()
c. selenium.type()
d. sendKeys("text")

b. sendKeys()

The sendKeys() method in WebDriver is used to enter values into a text box. It is the most commonly used method to input text into a text field. The method takes a string argument representing the text that you want to enter into the text box.

Question 8:

Is it possible to enable JavaScript in HTMLUnitDriver during driver initialization? If yes, then how?

a. WebDriver driver = new HtmlUnitDriver(true);
b. HtmlUnitDriver driver = new HtmlUnitDriver(true);
c. Not possible

a. WebDriver driver = new HtmlUnitDriver(true);

Yes, it is possible to enable JavaScript in HTMLUnitDriver during driver initialization. By default, JavaScript is disabled in HtmlUnitDriver. To enable JavaScript, you need to pass true as an argument when creating an instance of the HtmlUnitDriver. For example:

WebDriver driver = new HtmlUnitDriver(true);

This will create a HtmlUnitDriver instance with JavaScript enabled, which you can use to execute scripts on a web page. With JavaScript enabled, you can interact with the dynamic elements of a web page, such as drop-down menus, modal dialogs, and other JavaScript-based elements.

Question 9:

Which of the following correctly defines the behavior of the expression:

@Test(timeout=100)

a. The test will pass, if the method takes longer than 100 milliseconds.
b. The test will fail, if the method takes longer than 100 milliseconds.
c. It is same as @Test.
d. None

b. The test will fail, if the method takes longer than 100 milliseconds.

The timeout attribute in the @Test annotation sets a timeout value for the test method in milliseconds. If the test method takes longer than the specified timeout value to complete, it will fail and an exception will be thrown.

The test will fail if it takes longer than 100 milliseconds to complete. This is useful for detecting tests that are stuck in an infinite loop or are taking too long to complete for some other reason. By setting a timeout value, you can ensure that tests that are

taking too long to complete are terminated, and the results of the test can be reported to you.

Question 10:

Which is the only browser that does not support Selenium IDE to be used?

a. Firefox
b. Google Chrome
c. Safari
d. None of these

d. None of these

All major web browsers, including Firefox, Google Chrome, and Safari, support the use of Selenium IDE. Selenium IDE is a browser extension that provides a record-and-playback tool for creating and executing automated tests on web applications. It is compatible with all major web browsers, including Firefox, Google Chrome, and Safari.

However, it's worth noting that Selenium IDE is no longer actively maintained and is being replaced by other tools in the Selenium suite, such as Selenium WebDriver. While Selenium IDE may still be useful for simple test cases and learning Selenium, for more complex tests and automation, it's recommended to use other tools in the Selenium suite.

Question 11:

Which of these is NOT a feature of Selenium 4?

a. Relative Locators
b. WebElement Screenshot
c. Driver Constructors
d. Chrome DevTools

d. Chrome DevTools

Selenium 4 does not have a feature called Chrome DevTools. The features of Selenium 4 include:

a. Relative Locators: A new feature in Selenium 4 that allows you to locate elements on a web page relative to other elements. This makes it easier to locate elements on a page when their absolute position is not known.

b. WebElement Screenshot: A new feature in Selenium 4 that allows you to take a screenshot of a specific WebElement, rather than just the entire page.

c. Driver Constructors: A new feature in Selenium 4 that allows you to create drivers for different browsers in a more concise and readable way.

Chrome DevTools is not a feature of Selenium 4, but rather a feature of Google Chrome that allows you to inspect and debug web pages in the browser. However, Selenium can be used in conjunction with Chrome DevTools to automate tests that involve debugging and inspection.

Question 12:

Which of the following Selenium api calls you find best to complete the below statement?"_____ statement clicks on the first image that has an id attribute starting with 'lambdatest'."

a. selenium.click("//img[starts-with(@id,'lambdatest')]");
b. selenium.click("[starts-with(@id,'lambdatest')]");
c. selenium.click("//img[(@id,'lambdatest')]");
d. None

The correct Selenium API call to complete the statement "clicks on the first image that has an id attribute starting with 'lambdatest'" is:

a. selenium.click("//img[starts-with(@id,'lambdatest')]");

The XPath expression "//img[starts-with(@id,'lambdatest')]" matches all <img> elements that have an id attribute that starts with the string "lambdatest". The selenium.click method is used to simulate a click on the first element that matches the XPath expression.

Option b, selenium.click("[starts-with(@id,'lambdatest')]"), is not a correct XPath expression, as it only matches elements that have an id attribute that starts with the string "lambdatest", without specifying the type of element.

Option c, selenium.click("//img[(@id,'lambdatest')]"), is not a correct XPath expression, as it matches all <img> elements that have an id attribute equal to the string "lambdatest", instead of starting with the string "lambdatest".

Question 13:

What does the below Selenium WebDriver code would subject to do?

(Note:click on Question to enlarge)

a. This will create a new alert popup.
b. This will return the currently open alert popup.
c. This will close the currently open alert popup.
d. This will close all of the currently open alert popups.

b. This will return the currently open alert popup.

The code Alert alert = driver.switchTo().alert(); is used to return the currently open alert popup. The driver.switchTo().alert() method returns an Alert object, which represents an alert dialog in the browser. By storing the result of the method call in the alert variable, you can interact with the alert dialog, for example by calling alert.accept() to accept the alert, or alert.dismiss() to dismiss the alert.

Option a, This will create a new alert popup, is incorrect, as the code only returns an existing alert dialog, it does not create a new one.

Option c, This will close the currently open alert popup, is also incorrect, as the code only returns an Alert object, it does not close the alert dialog.

Option d, This will close all of the currently open alert popups, is also incorrect, as the code only returns a single Alert object for the currently open alert dialog, it does not close all alert dialogs.

Question 14:

Which of the following type of text pattern is used in Selenium?

a. Globbing
b. Unicode
c. Chronological Sequence
d. Simulation
e. None of these

a. Globbing.

*Globbing is a type of text pattern that uses wildcard characters, such as * or ?, to match multiple characters in a string. In Selenium, globbing is used to match the values of attributes, such as the text of an element, or the value of an input field. For example, you might use a globbing pattern to find all elements that have an id attribute that starts with a certain string:*

WebElement element = driver.findElement(By.xpath("//[starts-with(@id, 'lambdatest')]"));*

*In this example, the * wildcard character is used to match any element, and the starts-with function is used to match the id attribute based on the value.*

Question 15:

Which one is not correct about Selenium Grid?

a. To run tests in different browsers (except HtmlUnit) on different OS
b. To create tests with little or no prior knowledge in programming
c. To test a web application against Firefox only
d. To run a huge test suite, that can be executed in distributed environment

b. To create tests with little or no prior knowledge in programming.

Selenium Grid is a tool for running tests in a distributed environment, across multiple machines and browsers. It is not designed for creating tests, but for executing existing tests in parallel. In order to use Selenium Grid, you will need

some programming skills, as you will need to write tests using a programming language, such as Java or Python, and the Selenium WebDriver API. So, creating tests with little or no prior knowledge in programming is not a correct statement about Selenium Grid.

Question 16:

Which of the following syntax would you use to locate an element using inner text?

a. css=tag:contains(“inner text”)
b. css=tag:value(“inner text”)
c. css=tag:class(“inner text”)
d. css=tag:attributes(“inner text”)

a. css=tag:contains(“inner text”)

In Selenium, you can locate an element using its inner text by using the ":contains" pseudo-class in a CSS selector. The syntax would look something like this: "css=tag:contains('inner text')". This selector would match any tag (e.g. div, span, p, etc.) that has an inner text that contains the specified string "inner text". For example, you could locate a button with the text "Click Me" using the following selector: "css=button:contains('Click Me')". Note that this syntax is case-sensitive.

Question 17:

Which of the following commands would help to check the presence of a certain element?

a. verifyTable
b. verifyTitlePresent
c. verifyTextPresent
d. verifyElementPresent

The command to check the presence of a certain element in Selenium WebDriver is "verifyElementPresent". The other commands are not present in Selenium WebDriver.

Question 18:

Why would you like to choose WebDriver over Selenium RC?

a. WebDriver supports headless HTMLUnitDriver that enables faster test execution.
b. WebDriver does not need the Selenium RC server to be running.
c. WebDriver has native web browser support and runs faster than RC.
d. All of these

The correct answer is "d. All of these".

Selenium WebDriver is an improved version of Selenium RC and is widely preferred over Selenium RC. WebDriver offers several advantages over RC, including:

WebDriver supports headless HTMLUnitDriver that enables faster test execution.

WebDriver does not need the Selenium RC server to be running, making it easier to use and more efficient.

WebDriver has native web browser support and runs faster than RC, making it a more effective tool for automating web application testing.

All of these reasons make WebDriver a more popular choice for automating web application tests, and it is widely preferred over Selenium RC.

Question 19:

Which of the following is the correct difference between getWindowHandles() and getWindowHandle()?

a. getWindowHandles() returns a String whereas getWindowHandle() returns an Iterator< String >
b. getWindowHandles() returns the active browser handle whereas getWindowHandle() gives the top most browser handle
c. getWindowHandles() returns handles of all the open browsers whereas getWindowHandle() gets the address of the current browser.
d. None of these

The correct answer is "c. getWindowHandles() returns handles of all the open browsers whereas getWindowHandle() gets the address of the current browser."

In Selenium WebDriver, the getWindowHandles() method returns a set of window handles, which represent all the open browser windows or tabs. The returned set of window handles can be used to switch between the different open browser windows or tabs.

On the other hand, the getWindowHandle() method returns a string that represents the handle of the current browser window or tab. This method is typically used when you want to focus on a specific browser window or tab, and you have the handle of that window or tab.

In summary, getWindowHandles() returns handles of all the open browsers, whereas getWindowHandle() returns the handle of the current browser.

Question 20:

What does this regular expression match? Assume, you are using the number 1.

(Note:click on Question to enlarge)

a. 11,3,4
b. 11,11,111
c. 11,11,11
d. 11,111,1111

Question 21:

Which of the following WebDriver commands is used to check the presence of a web element?

a. verifyElementPresent
b. verifyTextPresent
c. isElementPresent
d. isElementExist

The command to check the presence of a web element in Selenium WebDriver is "isElementPresent". The other commands are not present in Selenium WebDriver.

The method isElementPresent returns a boolean value indicating whether the specified web element is present on the page or not.

Question 22:

Which of the following is correct in case of WebDriver?

a. WebDriver is an interface.
b. WebDriver is a class.

The correct answer is "a. WebDriver is an interface".

In Selenium WebDriver, the WebDriver interface is the primary interface for interacting with web browsers. The WebDriver interface provides a set of methods that can be used to automate various tasks on a web page, such as navigating to a URL, finding elements, clicking elements, and sending keyboard and mouse inputs. Several concrete implementations of the WebDriver interface are provided by the Selenium WebDriver API, including implementations for the major web browsers like Google Chrome, Mozilla Firefox, and Internet Explorer. To use Selenium WebDriver to automate tests, you create an instance of one of these concrete implementations, configure it as needed, and then use its methods to interact with the web page.

Question 23:

Which of the following WebDriver methods is used to change focus to an alert, a frame or a browser window?

a. switchTo()
b. changeTo()
c. setFocus()
d. changeFocus()

The correct answer is "a. switchTo()".

In Selenium WebDriver, the switchTo() method is used to change focus to an alert, a frame, or a browser window. The switchTo() method returns an instance of the

TargetLocator interface, which provides methods to switch between different windows, frames, and alerts on the web page.

Question 24:

What is the purpose of WebDriverBackedSelenium class in WebDriver application?

a. It's a WebDriver class implementing the Selenium-RC APIs.
b. It's a generic implementation of WebDriver to support multiple browsers.
c. It's a Selenium class to call WebDriver APIs.
d. None of these

The purpose of the WebDriverBackedSelenium class in a WebDriver application is "c. It's a Selenium class to call WebDriver APIs."

The WebDriverBackedSelenium class is a legacy class in the Selenium API that was introduced as a way to allow Selenium 1 (Selenium RC) tests to be executed using the new Selenium 2 (WebDriver) API. The WebDriverBackedSelenium class implements the Selenium interface and provides a bridge between the old Selenium RC API and the new WebDriver API.

By creating an instance of the WebDriverBackedSelenium class, you can use the Selenium 1 API to interact with the web page, and the WebDriver API will be used under the hood to perform the actual automation tasks. This can be useful if you have existing Selenium RC tests that you want to migrate to Selenium 2, as you can reuse the test code and update the automation code gradually.

However, it is recommended to migrate your tests to the new WebDriver API directly, as the Selenium RC API is now considered to be deprecated and may be removed in future versions of Selenium.

Question 25:

Select valid time out options in Selenium WebDriver

a. Implicit TimeOut

b. Explicit Time out
c. Fluent Timeout
d. Thread Time out

The valid timeout options in Selenium WebDriver are:

a. Implicit Timeout: An implicit timeout defines the amount of time the WebDriver will wait for an element to appear on the page before throwing a NoSuchElementException error. This timeout is applied globally for the entire WebDriver session, so it affects all elements that you interact with during that session.

b. Explicit Timeout: An explicit timeout is a method-level timeout that you can set for a specific WebDriver command, such as findElement() or findElements(). With an explicit timeout, you can control the amount of time that WebDriver will wait for a specific element to appear on the page before throwing a NoSuchElementException error.

c. Fluent Timeout: Fluent timeout is a time management concept used in Selenium to manage the wait time between multiple actions performed on a web page. It is used to wait for a specific amount of time before performing the next action, which helps to stabilize the tests.

Note: "c. Fluent Timeout" is not an official timeout option in Selenium WebDriver, but is a concept that is used in some implementations of the framework to manage timeouts.

So, the correct answers are: "a. Implicit Timeout" and "b. Explicit Timeout".

Question 26:

Which of the following is the recommended way to handle dynamic elements?

a. By using CSS locators.
b. By using relative xpath locators.
c. By using regular expressions.
d. All of these

d. All of these

The recommended way to handle dynamic elements in Selenium WebDriver can vary depending on the specific situation, but some common approaches include:

a. Using CSS locators: CSS locators are used to locate elements on a web page based on their CSS properties, such as their class, ID, tag name, etc. CSS locators are fast, reliable, and easy to use, making them a good choice for locating dynamic elements on a page.

b. Using relative xpath locators: Xpath is a language used to navigate XML documents, and it can also be used to locate elements in HTML pages. Relative xpath locators are a good choice for locating dynamic elements on a page because they can be based on a partial match of the element's properties, such as its tag name, attributes, and text content.

c. Using regular expressions: Regular expressions are a powerful tool for pattern matching and can be used to locate dynamic elements on a page by matching patterns in the element's properties, such as its ID or class.

In summary, there is no single "best" way to handle dynamic elements in Selenium WebDriver, and the approach that you choose will depend on the specific requirements of your test case. In many cases, a combination of these techniques may be used to achieve the desired result.

Question 27:

What does the below lines of code meant for?

a. Will simply start the browser and open the site.
b. Will run webdriver tests using selenium.
c. Will run Selenium 1.0 tests in webdriver.
d. None of these.

Question 28:

Which of the following correctly describes the difference between Thread.Sleep() and Selenium.setSpeed()?

a. Selenium.setSpeed() - takes a single argument in integer format. Thread.sleep() - takes a single argument in string format.
b. Selenium.setSpeed() - Runs first command after setSpeed delay by the number of milliseconds mentioned in setSpeed. Thread.sleep() - Waits for each command given after sleep.
c. Selenium.setSpeed() - Runs each command after setSpeed delay by the number of milliseconds specified in setSpeed(). Thread.sleep() - Waits for only once at the command given at sleep.

b. Selenium.setSpeed() - Runs each command after setSpeed delay by the number of milliseconds specified in setSpeed(). Thread.sleep() - Waits for only once at the command given at sleep.

setSpeed in Selenium is used to control the speed of test execution, where the delay between each command is set by the specified number of milliseconds. On the other hand, Thread.sleep is a method in Java that causes the current thread to wait for the specified number of milliseconds.

Question 29:

When the exception would occur in the below code snippet?

(Note:click on Question to enlarge)

a. Only If expectedValue and actualValue are same.
b. Only If expectedValue is different from the actualValue.
c. Only If actualValue is null and the expectedValue is non-null.
d. None of these

b. Only If expectedValue is different from the actualValue.

The exception would occur if the actual value (actualValue) is not equal to the expected value (expectedValue). In that case, an AssertionError will be thrown. This is a standard Java exception used to indicate that an assertion has failed.

Question 30:

Which of the following would you use to open an excel file for reading?

a. FileInputStream stream = FileInputStream(excelobj);
b. File excelobj = new File (); FileInputStream stream = new FileInputStream(excelobj);
c. FileInputStream stream = new FileInputStream(excelobj);
d. None

c. FileInputStream stream = new FileInputStream(excelobj);

The code creates a FileInputStream object, stream, that is used to read data from an excel file. The argument to the constructor of FileInputStream is the excel file object excelobj.

Option a is incorrect because it's missing the class name for FileInputStream, which should be new FileInputStream(excelobj).

Option b is incorrect because the File object is not initialized with any file path.

Option d is incorrect because one of the above options is the correct way to open an excel file for reading.

Question 31:

What exactly are the below instructions meant for?

a. Changing the User Agent.
b. Creating a FirefoxDriver object.
c. Creating a FirefoxProfile profile object.
d. None of these.

Question 32:

In Webdriver, which of the following methods navigates to a URL?

a. get("url")
b. getUrl("url")
c. navigate.to("url")
d. goToUrl("url")

The correct method to navigate to a URL in WebDriver is: driver.get("url")

Question 33:

What are the possible ways to take a screenshot using selenium WebDriver?

a. Full page using driver Object
b. Only selected element
c. It's not possible to take screenshots
d. Must user third party libraries to take screenshots

a. Full page using driver Object

b. Only selected element

Question 34:

Which of the following steps is not mandatory to launch Firefox in Selenium 3?

a. System.setProperty("webdriver.gecko.driver", "geckodriver path");
b. DesiredCapabilities caps = new DesiredCapabilities();
c. caps.setCapability("marionette", true);
d. WebDriver driver = new FirefoxDriver();
e. driver.get("http://www.google.com");

Step c, which sets the capability "marionette" to true, is not a mandatory step to launch Firefox in Selenium 3.

Question 35:

In Selenium terminology, what is the generic name for an argument that starts with "//"?

a. Element id
b. Partial link text
c. Xpath
d. Link Text
e. CSS

In Selenium terminology, the generic name for an argument that starts with "//" is Xpath.

Question 36:

WebDriver's Actions commands are <... complete the sentence ... >?

a. commands that directly interact with page elements.
b. commands that allow you to store values to a variable.
c. commands that verify if a certain condition is met.
d. commands that sends key strokes to browser.

a. commands that directly interact with page elements.

Question 37:

What is the contextClick() used for?

a. It is used to left click.
b. It is used to right click.
c. It is used to click the hidden element.
d. It is used to open popup menu.

b. It is used to right click. The contextClick() method is used to perform a right-click on an element in a web page using Selenium WebDriver.

Question 38:

Which of the following WebDriver method that supports moving between named windows?

a. driver.MoveTo().window("windowName");
b. driver.switchTo().window("windowName");
c. driver.Activate().window("windowName");
d. None of these

b. driver.switchTo().window("windowName");

Question 39:

How to check if a check box is checked or not?

a. driver.findElement(By.id(<>).Selected()

b. driver.findElement(By.id(<>).isSelected()
c. driver.findElement(By.id(<>).isChecked()
d. All of these

b. driver.findElement(By.id(<>).isSelected()

Question 40:

In Webdriver, which of the following commands returns the text of a html element?

a. getText(element)
b. getElementText()
c. getText()
d. selectText()
e. None

c. getText()