**Q #1) What is Automation Testing?**

Automation testing or Test Automation is a process of automating the manual process to test the application/system under test. Automation testing involves the use of a separate testing tool which lets you create test scripts which can be executed repeatedly and doesn’t require any manual intervention.

**Q #2) What are the benefits of Automation Testing?**

Benefits of Automation testing are:

1. Supports execution of repeated test cases
2. Aids in testing a large test matrix
3. Enables parallel execution
4. Encourages unattended execution
5. Improves accuracy thereby reducing human-generated errors
6. Saves time and money

**Q #3) Why should Selenium be selected as a test tool?**

Selenium

1. is a free and open source
2. have a large user base and helping communities
3. have cross Browser compatibility (Firefox, Chrome, Internet Explorer, Safari etc.)
4. have great platform compatibility (Windows, Mac OS, Linux etc.)
5. supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)
6. has fresh and regular repository developments
7. supports distributed testing

**Q #4) What is Selenium? What are the different Selenium components?**

Selenium is one of the most popular automated testing suites. Selenium is designed in a way to support and encourage automation testing of functional aspects of web-based applications and a wide range of browsers and platforms. Due to its existence in the open source community, it has become one of the most accepted tools amongst the testing professionals.

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason, it is referred to as a Suite. Each of these tools is designed to cater different testing and test environment requirements.

The suite package constitutes the following sets of tools:

* [**Selenium Integrated Development Environment (IDE)**](https://www.softwaretestinghelp.com/selenium-ide-download-and-installation-selenium-tutorial-2/) – Selenium IDE is a record and playback tool. It is distributed as a Firefox Plugin.
* **Selenium Remote Control (RC)** – Selenium RC is a server that allows a user to create test scripts in the desired programming language. It also allows executing test scripts within the large spectrum of browsers.
* [**Selenium WebDriver**](https://www.softwaretestinghelp.com/selenium-webdriver-selenium-tutorial-8/) – WebDriver is a different tool altogether that has various advantages over Selenium RC. WebDriver directly communicates with the web browser and uses its native compatibility to automate.
* [**Selenium Grid**](https://www.softwaretestinghelp.com/selenium-grid-selenium-tutorial-29/) – Selenium Grid is used to distribute your test execution on multiple platforms and environments concurrently.

**Q #5) What are the testing types that can be supported by Selenium?**

Selenium supports the following types of testing:

1. Functional Testing
2. Regression Testing

**Q #6) What are the limitations of Selenium?**

Following are the limitations of Selenium:

* Selenium supports testing of only web-based applications
* Mobile applications cannot be tested using Selenium
* Captcha and Barcode readers cannot be tested using Selenium
* Reports can only be generated using third-party tools like TestNG or JUnit.
* As Selenium is a free tool, thus there is no ready vendor support through the user can find numerous helping communities.
* The user is expected to possess prior programming language knowledge.

**Q #10)** **What are the different types of locators in Selenium?**

The locator can be termed as an address that identifies a web element uniquely within the webpage. Thus, to identify web elements accurately and precisely we have [different types of locators in Selenium](https://www.softwaretestinghelp.com/using-selenium-xpath-and-other-locators-selenium-tutorial-5/):

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

**Q #11)** **What is the difference between assert and verify commands?**

**Assert:**Assert command checks whether the given condition is true or false. Let’s say we assert whether the given element is present on the web page or not. If the condition is true then the program control will execute the next test step but if the condition is false, the execution would stop and no further test would be executed.

**Verify:**Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn’t halt i.e. any failure during verification would not stop the execution and all the test steps would be executed.

**Q #12) What is an XPath?**

[XPath](https://www.softwaretestinghelp.com/using-selenium-xpath-and-other-locators-selenium-tutorial-5/) is used to locate a web element based on its XML path. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data. It stores data in a key-value pair which is very much similar to HTML tags. Both being markup languages and since they fall under the same umbrella, XPath can be used to locate HTML elements.

The fundamental behind locating elements using XPath is the traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

**Q #13) What is the difference between “/” and “//” in Xpath?**

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

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

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

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

**Q #19) What are the different types of Drivers available in WebDriver?**

The different drivers available in WebDriver are:

* FirefoxDriver
* InternetExplorerDriver
* ChromeDriver
* SafariDriver
* OperaDriver
* AndroidDriver
* IPhoneDriver
* HtmlUnitDriver

**Q #20) What are the different types of waits available in WebDriver?**

There are two [types of waits available in WebDriver](https://www.softwaretestinghelp.com/selenium-webdriver-waits-selenium-tutorial-15/):

1. Implicit Wait
2. Explicit Wait

**Implicit Wait:**Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, the subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.

**Explicit Wait:** Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, explicit waits are applied for a particular instance only.

**Q #21)** **How to type in a textbox using Selenium?**

The user can use sendKeys(“String to be entered”) to enter the string in the textbox.

**Syntax:**  
*WebElement username = drv.findElement(By.id(“Email”));*  
*// entering username*  
*username.sendKeys(“sth”);*

**Q #22)** **How can you find if an element in displayed on the screen?**

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

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

**Syntax:**

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

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

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

**Q #23)** **How can we get a text of a web element?**

Get command is used to retrieve the inner text of the specified web element. The command doesn’t require any parameter but returns a string value. It is also one of the extensively used commands for verification of messages, labels, errors etc displayed on the web pages.

**Syntax:**  
*String Text = driver.findElement(By.id(“Text”)).getText();*

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

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

**Syntax:**

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

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

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

**Q #25) What are the different types of navigation commands?**

Following are the [navigation commands](https://www.softwaretestinghelp.com/selenium-webdriver-waits-selenium-tutorial-15/):  
**navigate().back()** – The above command requires no parameters and takes back the user to the previous webpage in the web browser’s history.

**Sample code:**  
*driver.navigate().back();*

**navigate().forward()** – This command lets the user to navigate to the next web page with reference to the browser’s history.

**Sample code:**  
*driver.navigate().forward();*

**navigate().refresh()** – This command lets the user to refresh the current web page there by reloading all the web elements.

**Sample code:**  
*driver.navigate().refresh();*

**navigate().to()** – This command lets the user to launch a new web browser window and navigate to the specified URL.

**Sample code:**  
*driver.navigate().to(“https://google.com”);*

**Q #26) How to click on a hyper link using linkText?**

*driver.findElement(By.linkText(“Google”)).click();*

The command finds the element using link text and then click on that element and thus the user would be re-directed to the corresponding page.

The above-mentioned link can also be accessed by using the following command.

*driver.findElement(By.partialLinkText(“Goo”)).click();*

The above command finds the element based on the substring of the link provided in the parenthesis and thus partialLinkText() finds the web element with the specified substring and then clicks on it.

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

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

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

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

**Locating iframe using index**

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

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

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

**Q #28) When do we use findElement() and findElements()?**

**findElement():**findElement() is used to find the first element in the current web page matching to the specified locator value. Take a note that only first matching element would be fetched.

**Syntax:**

*WebElement element = driver.findElements(By.xpath(“//div[@id='example']//ul//li”));*  
**findElements():**findElements() is used to find all the elements in the current web page matching to the specified locator value. Take a note that all the matching elements would be fetched and stored in the list of WebElements.

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

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

At times, we may come across elements of the same type like multiple hyperlinks, images etc arranged in an ordered or unordered list. Thus, it makes absolute sense to deal with such elements by a single piece of code and this can be done using WebElement List.

**Sample Code**

[view source](https://www.softwaretestinghelp.com/selenium-interview-questions-answers/#viewSource)

[print](https://www.softwaretestinghelp.com/selenium-interview-questions-answers/#printSource)[?](https://www.softwaretestinghelp.com/selenium-interview-questions-answers/#about)

|  |  |
| --- | --- |
| 1 | // Storing the list |
| 2 | List <WebElement> elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li")); | |

|  |  |
| --- | --- |
| 3 | // Fetching the size of the list |
| 4 | int listSize = elementList.size(); | |

|  |  |  |
| --- | --- | --- |
| 5 | for (int i=0; i<listSize; i++) | |
| 6 | { |

|  |  |  |
| --- | --- | --- |
| 7 | // Clicking on each service provider link | |
| 8 | serviceProviderLinks.get(i).click(); |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | // Navigating back to the previous page that stores link to service providers | | |
| 10 | | driver.navigate().back(); |

|  |  |
| --- | --- |
| 11 | } |

**Q #30) What is the difference between driver.close() and driver.quit command?**

**close()**: WebDriver’s close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does it return any value.

**quit()**: Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does is return any value.

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

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

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

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

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

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

**Q #33) How can we handle windows based pop up?**

Selenium is an automation testing tool which supports only web application testing, that means, it doesn’t support testing of windows based applications. However Selenium alone can’t help the situation but along with some third-party intervention, this problem can be overcome. There are several third-party tools available for handling window based pop-ups along with the selenium like AutoIT, Robot class etc.

**Q #34) How to assert the title of the web page?**

*//verify the title of the web page*  
*assertTrue(“The title of the window is incorrect.”,driver.getTitle().equals(“Title of the page”));*

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

WebDriver offers a wide range of interaction utilities that the user can exploit to automate mouse and keyboard events. Action Interface is one such utility which simulates the single user interactions.

Thus, In the following scenario, we have used Action Interface to mouse hover on a drop down which then opens a list of options.

**Sample Code:**

[view source](https://www.softwaretestinghelp.com/selenium-interview-questions-answers/#viewSource)

[print](https://www.softwaretestinghelp.com/selenium-interview-questions-answers/#printSource)[?](https://www.softwaretestinghelp.com/selenium-interview-questions-answers/#about)

|  |  |
| --- | --- |
| 1 | // Instantiating Action Interface |
| 2 | Actions actions=new Actions(driver); | |

|  |  |
| --- | --- |
| 3 | // howering on the dropdown |
| 4 | actions.moveToElement(driver.findElement(By.id("id of the dropdown"))).perform(); | |

|  |  |
| --- | --- |
| 5 | // Clicking on one of the items in the list options |
| 6 | WebElement subLinkOption=driver.findElement(By.id("id of the sub link")); | |

|  |  |
| --- | --- |
| 7 | subLinkOption.click(); |

**Q #36) How to retrieve CSS properties of an element?**

The values of the css properties can be retrieved using a get() method:

**Syntax:**  
*driver.findElement(By.id(“id“)).getCssValue(“name of css attribute”);*  
*driver.findElement(By.id(“id“)).getCssValue(“font-size”);*

**Q #38) What is Junit?**

[Junit](https://www.softwaretestinghelp.com/selenium-junit-framework-selenium-tutorial-11/) is a unit testing framework introduced by Apache. Junit is based on Java.

**Q #39) What are Junit annotations?**

Following are the JUnit Annotations:

* **@Test:**Annotation lets the system know that the method annotated as @Test is a test method. There can be multiple test methods in a single test script.
* **@Before:**Method annotated as @Before lets the system know that this method shall be executed every time before each of the test methods.
* **@After:**Method annotated as @After lets the system know that this method shall be executed every time after each of the test method.
* **@BeforeClass:**Method annotated as @BeforeClass lets the system know that this method shall be executed once before any of the test methods.
* **@AfterClass:**Method annotated as @AfterClass lets the system know that this method shall be executed once after any of the test methods.
* **@Ignore:**Method annotated as @Ignore lets the system know that this method shall not be executed.

**Q #40)** **What is TestNG and how is it better than Junit?**

[TestNG](https://www.softwaretestinghelp.com/testng-framework-selenium-tutorial-12/) is an advanced framework designed in a way to leverage the benefits by both the developers and testers. With the commencement of the frameworks, JUnit gained enormous popularity across the Java applications, Java developers and Java testers with remarkably increasing the code quality. Despite being easy to use and straightforward, JUnit has its own limitations which give rise to the need of bringing TestNG into the picture. TestNG is an open source framework which is distributed under the Apache Software License and is readily available for download.

TestNG with WebDriver provides an efficient and effective test result format that can, in turn, be shared with the stakeholders to have a glimpse on the product’s/application’s health thereby eliminating the drawback of WebDriver’s incapability to generate test reports. TestNG has an inbuilt exception handling mechanism which lets the program to run without terminating unexpectedly.

There are various advantages that make TestNG superior to JUnit. Some of them are:

* Added advance and easy annotations
* Execution patterns can set
* Concurrent execution of test scripts
* Test case dependencies can be set

**Q #42) What is a framework?**

The framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing.

**Q #43)** **What are the advantages of the Automation framework?**

**The advantage of**[**Test Automation framework**](https://www.softwaretestinghelp.com/test-automation-frameworks-selenium-tutorial-20/)

* Reusability of code
* Maximum coverage
* Recovery scenario
* Low-cost maintenance
* Minimal manual intervention
* Easy Reporting