**1. What is Automation Testing?**

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

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

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

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

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

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

1. We cannot test windows application
2. We cannot test mobile apps
3. Limited reporting
4. Handling dynamic Elements
5. Handling page load
6. Handling pop up windows
7. Handling captcha

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

It is one of the Selenium Tricky Interview Questions.

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

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

I prefer Selenium Automation Tool because some of the benefits of Selenium to do automation testing are

* **Free and open source –** It is a free open source tool. There is no need to allot budget for this tool
* **Help –** Have large user base and helping communities.
* [**Cross-browser compatibility**](https://www.softwaretestingmaterial.com/cross-browser-testing/)**–** It works on almost all popular browsers such as Chrome, Firefox, Internet Explorer, and Safari.
* **Cross Platform compatibility –** It works on platforms such as Windows, Linux, Mac.
* **Multiple programming languages –** It supports programming languages such as [Java](https://www.softwaretestingmaterial.com/java-tutorial/), Phyton, Perl, Php, C#, Ruby, etc.,
* **Parallel Execution –** Selenium Grid supports parallel execution of Selenium Scripts.
* **Continuous Integration –** We can achieve nightly execution using Jenkins.

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

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

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

The fastest implementation of WebDriver is the HTMLUnitDriver. It is because the HTMLUnitDriver does not execute tests in the browser. Starting a browser and running test cases took more time compared to running the scripts without a browser. HTMLUnitDriver took a simple HTTP request-response mechanism for test case execution.

**25. What are the Operating Systems supported by Selenium WebDriver?**

* Windows
* Linux
* Mac OS X
* iOS
* Android

26. What are the testing types that can be supported by selenium?

Testing types that can be supported by Selenium are as follows:

* Functional Testing
* Regression Testing
* Retesting
* Acceptance Testing
* End-to-End Testing
* Smoke Testing
* Sanity Testing
* Responsive Testing
* Cross Browser Testing
* UI Testing
* Integration Testing

27. How many parameters can selenium commands have at minimum?

There are four parameters that you have to pass in Selenium are

* Host
* Port Number
* Browser
* URL

**Host:** It is the parameter which we use to bind Selenium to a specific IP. Usually, we run selenium tests on our local machine so the value will be ‘localhost’. You can sepcify IP address instead of localhost.

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

**Port Number:** TCP/IP port which is used to connect selenium tests to the selenium grid hub. Default port hub is 4444.  
java -jar <selenium server standalone jar name> -role hub -port 4444

Make sure no other application in your system is using this port. You may face an exception like Exception in thread “main” java.net.BindException: Selenium is already running on port 4444. Or some other service is.

If this occurs you can either shutdown the other process that is using port 4444, or you can tell Selenium-Grid to use a different port for its hub. Use the -port option for changing the port used by the hub.

java -jar <selenium server standalone jar name> -role hub -port 4441

**Browser:** To pass the browser which has to execute our selenium scripts

**URL:** To pass the application URL

**28. What are the Open-source Frameworks supported by Selenium WebDriver?**

* JUnit
* TestNG

### 31. When you use these locators ID, Name, XPath, Or CSS Selector?

**ID** & **Name** locators will be used when there are unique identifiers & unique names available on the web page.  
**CSS Selector** can be used for performance and when ID & Name locators are not unique.  
**XPath** is used when there is no preferred locators.

### ****32. What is the difference between “/” and “//”****

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

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

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

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



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

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



|  |  |
| --- | --- |
| 1 | //input[@id='email'] |

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

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

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

For detailed post check the below link.

**37. What are the verification points available in Selenium?**

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

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

**44. What are the different exceptions you have faced in Selenium WebDriver?**

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

1. ElementNotVisibleException
2. StaleElementReferenceException

**Element Not visible Exception:**

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

**Stale Element Reference Exception:**

A [stale element reference exception](https://www.softwaretestingmaterial.com/stale-element-reference-exception-selenium-webdriver/) is thrown in one of two cases, the first being more common than the second.

The two reasons for Stale element reference are

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

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

Some other exceptions we usually face are as follows:

* WebDriverException
* IllegalStateException
* TimeoutException
* NoAlertPresentException
* NoSuchWindowException
* NoSuchElementException

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

To do this we pass username and password with the URL

|  |  |
| --- | --- |
| 2 | http://username:password@url  e.g. http://myUserName:myPassword@softwaretestingmaterial.com |

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

Alternative method to driver.get(“url”) method is driver.navigate.to(“url”)

### ****65. What is the difference between****driver.get() ****and**** driver.navigate.to(“url”)****?****

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

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

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

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

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

There are multiple ways to refresh a page in selenium

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

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

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

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

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

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

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

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

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

### ****78. How to capture Screenshot in Selenium WebDriver?****

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

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

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

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

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

### ****1. How can we handle windows based pop up?****

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

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

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

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

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

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

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

### 89. Can You Use Selenium For Rest API Testing Or Web Services Testing?

Simple answer for this is Selenium is not a tool for API Testing. It automates web browsers. Rest API & Web Services contains no UI. So we cannot automate using Selenium.

### ****94. What is Page Object Model in Selenium?****

[Page Object Model](https://www.softwaretestingmaterial.com/page-object-model/) is a Design Pattern which has become popular in Selenium Test Automation. It is widely used design pattern in Selenium for enhancing test maintenance and reducing code duplication. Page object model (POM) can be used in any [kind of framework](https://www.softwaretestingmaterial.com/types-test-automation-frameworks/) such as modular, [data-driven](https://www.softwaretestingmaterial.com/data-driven-framework-selenium-webdriver/), keyword driven, hybrid framework etc.  A page object is an object-oriented class that serves as an interface to a page of your Application Under Test(AUT). The tests then use the methods of this page object class whenever they need to interact with the User Interface (UI) of that page. The benefit is that if the UI changes for the page, the tests themselves don’t need to change, only the code within the page object needs to change. Subsequently, all changes to support that new UI is located in one place.

### ****95. What is Page Factory?****

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

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

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

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

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

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

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

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

### ****115. What is TestNG?****

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to [integration testing](https://www.softwaretestingmaterial.com/integration-testing/).

### ****116. What are the annotations available in TestNG?****

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

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

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

Some of the common assertions supported by TestNG are

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