**Top 35 Selenium Interview Questions And Answers**

The entire set of Selenium interview questions, is divided into three sections:

1. [Basic level](https://www.edureka.co/blog/interview-questions/selenium-interview-questions-answers/?utm_source=youtube&utm_medium=youtube-description&utm_campaign=selenium-interview-questions-video" \l "BasicLevel)
2. [Advanced level](https://www.edureka.co/blog/interview-questions/selenium-interview-questions-answers/?utm_source=youtube&utm_medium=youtube-description&utm_campaign=selenium-interview-questions-video#AdvanceLevel)
3. [TestNG framework for Selenium](https://www.edureka.co/blog/interview-questions/selenium-interview-questions-answers/?utm_source=youtube&utm_medium=youtube-description&utm_campaign=selenium-interview-questions-video#TestNGFramework)

**A. Basic Level  – Selenium Interview Questions**

**1. What are the advantages and disadvantages of Selenium over other testing tools like QTP and RFT?**

The advantages of Selenium over QTP and RFT are:

**License**: Selenium is open source, whereas HP’s QTP and IBM’s RFT are licensed software.  
**Environment Support**: Selenium supports Windows OS, Linux OS, Solaris OS X (If browser & JVM or JavaScript support exists), whereas QTP and RFT work only on Windows OS.  
**Programming Language Support**: Selenium supports Java, C#, Ruby, Python, Perl, PHP and JavaScript, whereas RFT supports only Java and C# and QTP supports only VBScript.  
**Hardware resource consumption during script execution**: Selenium consumes very less hardware resource, but QTP and RFT a lot of hardware resource.  
**Coding experience**: For Selenium, coding skills should be very good along with technical capabilities of integrating the framework, whereas coding experience and skills are not that much needed for QTP and RFT.

**2. What are the significant changes in upgrades in various Selenium versions?**

Selenium v1 included only three suite of tools: Selenium IDE, Selenium RC and Selenium Grid. Note that there was no WebDriver in Selenium v1. Selenium WebDriver was introduced in Selenium v2. With the onset of WebDriver, Selenium RC got deprecated and is not in use since. Older versions of RC is available in the market though, but support for RC is not available. Currently, Selenium v3 is in use, and it comprises of IDE, WebDriver and Grid.

IDE is used for recording and playback of tests, WebDriver is used for testing dynamic web applications via a programming interface and Grid is used for deploying tests in remote host machines.

**3. Explain the different exceptions in Selenium WebDriver.**

Exceptions in Selenium are similar to exceptions in other programming languages. The most common exceptions in Selenium are:  
**TimeoutException:** This exception is thrown when a command performing an operation does not complete in the stipulated time.  
**NoSuchElementException:** This exception is thrown when an element with given attributes is not found on the web page.  
**ElementNotVisibleException:** This exception is thrown when the element is present in DOM (Document Object Model), but not visible on the web page.  
**StaleElementException:** This exception is thrown when the element is either deleted or no longer attached to the DOM.

**4. What is exception test in Selenium?**

An exception test is an exception that you expect will be thrown inside a test class. If you have written a test case in such way that it should throw an exception, then you can use the **@Test** annotation and specify which exception you will be expecting by mentioning it in the parameters. Take a look at the example below: **@Test(expectedException = NoSuchElementException.class)**

Do note the syntax, where the exception is suffixed with .class

**5. Why and how will you use an Excel Sheet in your project?**

The reason we use Excel sheets is because it can be used as data source for tests. An excel sheet can also be used to store the data set while performing DataDriven Testing. These are the two main reasons for using Excel sheets.

When you use the excel sheet as **data source**, you can store the following:

* **Application URL for all environments**: You can specify the URL of the environment in which you want to do the testing like: development environment or testing environment or QA environment or staging environment or production/ pre-production environment.
* **User name and password credentials of different environments:**You can store the access credentials of the different applications/ environments in the excel sheet. You can store them in encoded format and whenever you want to use them, you can decode them instead of leaving it plain and unprotected.
* **Test cases to be executed**: You can list down the entire set of test cases in a column and in the next column, you can specify either Yes or No which indicates if you want that particular test case to be executed or ignored.

When you use the excel sheet for **DataDriven Test**, you can store the data for different iterations to be performed in the tests. For example while testing a web page, the different sets of input data that needs to be passed to the test box can be stored in the excel sheet.

**6. How can you redirect browsing from a browser through some proxy?**

Selenium provides a PROXY class to redirect browsing from a proxy. Look at the example below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | String PROXY = “199.201.125.147:8080”;    org.openqa.selenium.Proxy proxy = new.org.openqa.selenium.Proxy();  proxy.setHTTPProxy(Proxy)   .setFtpProxy(Proxy)   .setSslProxy(Proxy)  DesiredCapabilities cap = new DesiredCapabilities();  cap.setCapability(CapabilityType.PROXY, proxy);  WebDriver driver = new FirefoxDriver(cap); |

**7. What is POM (Page Object Model)? What are its advantages?**

Page Object Model is a design pattern for creating an Object Repository for web UI elements. Each web page in the application is required to have it’s own corresponding page class. The page class is thus responsible for finding the WebElements in that page and then perform operations on those WebElements.

The advantages of using POM are:

* Allows us to separate operations and flows in the UI from Verification – improves code readability
* Since the Object Repository is independent of Test Cases, multiple tests can use the same Object Repository
* Reusability of code

**8. What is Page Factory?**

Page Factory gives an optimized way to implement Page Object Model. When we say it is optimized, it refers to the fact that the memory utilization is very good and also the implementation is done in an object oriented manner.

Page Factory is used to initialize the elements of the Page Object or instantiate the Page Objects itself. Annotations for elements can also be created (and recommended) as the describing properties may not always be descriptive enough to differentiate one object from the other.

The concept of separating the Page Object Repository and Test Methods is followed here also. Instead of having to use ‘FindElements’, we use annotations like:**@FindBy** to find WebElement, and **initElements** method to initialize web elements from the Page Factory class.

**@FindBy** can accept **tagName**, **partialLinkText**, **name**, **linkText**, **id**, **css**, **className**& **xpath**as attributes.

**9. What are the different types of WAIT statements in Selenium WebDriver? *Or the question can be framed like this:* How do you achieve synchronization in WebDriver?**

There are basically two types of wait statements: **Implicit Wait** and **Explicit Wait**.

Implicit wait instructs the WebDriver to wait for some time by polling the DOM. Once you have declared implicit wait, it will be available for the entire life of the WebDriver  instance. By default, the value will be 0. If you set a longer default, then the behavior will poll the DOM on a periodic basis depending on the browser/ driver implementation.

Explicit wait instructs the execution to wait for some time until some condition is achieved. Some of those conditions to be attained are:

* elementToBeClickable
* elementToBeSelected
* presenceOfElementLocated

**10. Write a code to wait for a particular element to be visible on a page. Write a code to wait for an alert to appear.**

We can write a code such that we specify the XPath of the web element that needs to be visible on the page and then ask the WebDriver to wait for a specified time. Look at the sample piece of code below:

|  |  |
| --- | --- |
| 1  2 | WebDriverWait wait=new WebDriverWait(driver, 20);  Element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( “<xpath”))); |

Similarly, we can write another piece of code asking the WebDriver to wait until an error appears like this:

|  |  |
| --- | --- |
| 1  2 | WebDriverWait wait=new WebDriverWait(driver, 20);  Element = wait.until(ExpectedConditions.alertIsPresent()); |

**11. What is the use of JavaScriptExecutor?**

**JavaScriptExecutor** is an interface which provides a mechanism to execute Javascript through the Selenium WebDriver. It provides “**executescript**” and “**executeAsyncScript**” methods, to run JavaScript in the context of the currently selected frame or window. An example of that is:

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor js = (JavascriptExecutor) driver;  js.executeScript(Script,Arguments); |

**12. How to scroll down a page using JavaScript in Selenium?**

We can scroll down a page by using window.scrollBy() function. Example:

|  |  |
| --- | --- |
| 1 | ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,500)"); |

**13. How to scroll down to a particular element?**

To scroll down to a particular element on a web page, we can use the function **scrollIntoView()**. Example:

|  |  |
| --- | --- |
| 1 | ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", element); |

**14. How to handle keyboard and mouse actions using Selenium?**

We can handle special keyboard and mouse events by using **Advanced User Interactions API**. The Advanced User Interactions API contains the Actions and the Action Classes that are needed for executing these events. Most commonly used keyboard and mouse events provided by the Actions class are in the table below:

|  |  |
| --- | --- |
| **Method** | **Description** |
| clickAndHold() | Clicks (without releasing) the current mouse location. |
| dragAndDrop() | Performs click-and-hold at the location of the source element, moves. |
| source, target() | Moves to the location of the target element, then releases the mouse. |

**15. What are different types of frameworks?**

The different types of frameworks are:

* Data Driven Framework:-  
  When the entire test data is generated from some external files like Excel, CSV, XML or some database table, then it is called Data Driven framework.
* Keyword Driven Framework:-  
  When only the instructions and operations are written in a different file like an Excel worksheet, it is called Keyword Driven framework.
* Hybrid Framework:-  
  A combination of both the Data Driven framework and the Keyword Driven framework is called Hybrid framework.

**16. Which files can be used as data source for different frameworks?**

Some of the file types of the dataset can be: excel, xml, text, csv, etc.

**17. How can you fetch an attribute from an element? How to retrieve typed text from a textbox?**

We can fetch the attribute of an element by using the **getAttribute()** method. Sample code:

|  |  |
| --- | --- |
| 1  2 | WebElement eLogin = driver.findElement(By.name(“Login”);  String LoginClassName = eLogin.getAttribute("classname"); |

Here, I am finding the web page’s login button named ‘Login’. Once that element is found, getAttribute() can be used to retrieve any attribute value of that element and it can be stored it in string format. In my example, I have retrieved ‘classname’ attribute and stored it in LoginClassName.

Similarly, to retrieve some text from any textbox, we can use getText() method. In the below piece of code I have retrieved the text typed in the ‘Login’ element.

|  |  |
| --- | --- |
| 1  2 | WebElement eLogin = driver.findElement(By.name(“Login”);  String LoginText = Login.getText (); |

In the below Selenium WebDriver tutorial, there is a detailed demonstration of locating elements on the web page using different element locator techniques and the basic methods/ functions that can be applied on those elements. 

**Selenium WebDriver Tutorial For Beginners | Edureka**

**B. Advanced Level – Selenium Interview Question**

**18. How to send ALT/SHIFT/CONTROL key in Selenium WebDriver?**

When we generally use ALT/SHIFT/CONTROL keys, we hold onto those keys and click other buttons to achieve the special functionality. So it is not enough just to specify **keys.ALT** or **keys.SHIFT** or **keys.CONTROL** functions.

For the purpose of holding onto these keys while subsequent keys are pressed, we need to define two more methods: **keyDown(modifier\_key)** and**keyUp(modifier\_key)**

Parameters: **Modifier\_key (keys.ALT or Keys.SHIFT or Keys.CONTROL)**Purpose: Performs a modifier key press and does not release the modifier key. Subsequent interactions may assume it’s kept pressed.

Parameters: **Modifier\_key (keys.ALT or Keys.SHIFT or Keys.CONTROL)**  
Purpose: Performs a key release.  
Hence with a combination of these two methods, we can capture the special function of a particular key.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | public static void main(String[] args)  {  String baseUrl = “https://www.facebook.com”;  WebDriver driver = new FirefoxDriver();    driver.get("baseUrl");  WebElement txtUserName = driver.findElement(By.id(“Email”);    Actions builder = new Actions(driver);  Action seriesOfActions = builder   .moveToElement(txtUerName)   .click()   .keyDown(txtUserName, Keys.SHIFT)   .sendKeys(txtUserName, “hello”)   .keyUp(txtUserName, Keys.SHIFT)   .doubleClick(txtUserName);   .contextClick();   .build();  seriesOfActions.perform();  } |

**19. How to take screenshots in Selenium WebDriver?**

You can take a screenshot by using the **TakeScreenshot** function. By using **getScreenshotAs()** method you can save that screenshot. Example:

|  |  |
| --- | --- |
| 1 | File scrFile = ((TakeScreenshot)driver).getScreenshotAs(outputType.FILE); |

**20. How to set the size of browser window using Selenium?**

To maximize the size of browser window, you can use the following piece of code:  
driver.manage().window().maximize(); – To maximize the window

To resize the current window to a particular dimension, you can use the **setSize()** method. Check out the below piece of code:

|  |  |
| --- | --- |
| 1  2  3 | System.out.println(driver.manage().window().getSize());  Dimension d = new Dimension(420,600);  driver.manage().window().setSize(d); |

To set the window to a particular size, use **window.resizeTo()** method. Check the below piece of code:

|  |  |
| --- | --- |
| 1 | ((JavascriptExecutor)driver).executeScript("window.resizeTo(1024, 768);"); |

To witness a demonstration on setting custom sizes for the browser window and finding various elements on the web page, see the video below.

**Selenium Tutorial For Beginners | Edureka**

**21. How to handle a dropdown in Selenium WebDriver? How to select a value from dropdown?**

Questions on dropdown and selecting a value from that dropdown are very common Selenium interview questions because of the technicality involved in writing the code.

The most important detail you should know is that to work with a dropdown in Selenium, we must always make use of this html tag: **‘select’**. Without using ‘select’, we cannot handle dropdowns. Look at the snippet below in which I have written a code for a creating a dropdown with three options.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <select id="mySelect">  <option value="option1">Cars</option>  <option value="option2">Bikes</option>  <option value="option3">Trains</option>  </select> |

In this code we use **‘select’**tag to define a dropdown element and the **id**of the dropdown element is ‘myselect’. We have 3 options in the dropdown: Cars, Bikes and Trains. Each of these options, have a ‘value’ attribute also assigned to them. First option from dropdown has value assigned as ‘option1’, second option has value = ‘option2’ and similarly third option has value assigned as ‘option3’.

If you are clear with the concept so far, then you can proceed to the next aspect of choosing a value from the dropdown. This is a 2 step process:

1. Identify the ‘select’ html element (Because dropdowns must have the ‘select’ tag)
2. Select an option from that dropdown element

To identify the ‘select’ html element from the web page, we need to use findElement() method. Look at the below piece of code:

|  |  |
| --- | --- |
| 1  2 | WebElement mySelectElement = driver.findElement(By.id("mySelect"));  Select dropdown = new Select(mySelectElement); |

Now to select an option from that dropdown, we can do it in either of the three ways:

1. dropdown.selectByVisibleText(“Bikes”); → Selecting an option by the text that is visible
2. dropdown.selectByIndex(“1”); → Selecting, by choosing the Index number of that option
3. dropdown.selectByValue(“option2”); → Selecting, by choosing the value of that option

Note that from the above example, in all the three cases, “Bikes” will be chosen from the dropdown. In the first case, we are choosing by visible text on the web page. When it comes to selection by index, 1 represents “Bikes” because indexing values start from 0 and then get incremented to 1 and 2. Finally in case of selection by value attribute, ‘option2’ refers to “Bikes”. So, these are the different ways to choose a value from a dropdown.

**22. How to switch to a new window (new tab) which opens up after you click on a link?**

If you click on a link in a web page, then for changing the WebDriver’s focus/ reference to the new window we need to use the **switchTo()** command. Look at the below example to switch to a new window:  
**driver.switchTo().window();**

Here, ‘windowName’ is the name of the window you want to switch your reference to.

In case you do not know the name of the window, then you can use the **driver.getWindowHandle()** command to get the name of all the windows that were initiated by the WebDriver. Note that it will not return the window names of browser windows which are not initiated by your WebDriver.

Once you have the name of the window, then you can use an enhanced for loop to switch to that window. Look at the piece of code below.

|  |  |
| --- | --- |
| 1  2  3  4  5 | String handle= driver.getWindowHandle();  for (String handle : driver.getWindowHandles())  {  driver.switchTo().window(handle);  } |

**23. How do you upload a file using Selenium WebDriver?**

To upload a file we can simply use the command **element.send\_keys(file path).**But there is a prerequisite before we upload the file. We have to use the html tag: **‘input’**and attribute type should be **‘file’**. Take a look at the below example where we are identifying the web element first and then uploading the file.

|  |  |
| --- | --- |
| 1  2  3 | <input type="file" name="uploaded\_file" size="50" class="pole\_plik">  element = driver.find\_element\_by\_id(”uploaded\_file")  element.send\_keys("C:\myfile.txt") |

**24. Can we enter text without using sendKeys()?**

Yes. We can enter/ send text without using **sendKeys()** method. We can do it using JavaScriptExecutor.

How do we do it?  
Using DOM method of, identification of an element, we can go to that particular document and then get the element by its ID (here login) and then send the text by value. Look at the sample code below:

|  |  |
| --- | --- |
| 1  2 | JavascriptExecutor jse = (JavascriptExecutor) driver;  jse.executeScript("document.getElementById(‘Login').value=Test text without sendkeys"); |

**25. Explain how you will login into any site if it is showing any authentication popup for username and password?**

Since there will be popup for logging in, we need to use the explicit command and verify if the alert is actually present. Only if the alert is present, we need to pass the username and password credentials. The sample code for using the explicit wait command and verifying the alert is below:

|  |  |
| --- | --- |
| 1  2  3 | WebDriverWait wait = new WebDriverWait(driver, 10);  Alert alert = wait.until(ExpectedConditions.alertIsPresent());  alert.authenticateUsing(new UserAndPassword(\*\*username\*\*, \*\*password\*\*)); |

**26. Explain how can you find broken links in a page using Selenium WebDriver?**

This is a trick question which the interviewer will present to you. He can provide a situation where in there are 20 links in a web page, and we have to verify which of those 20 links are working and how many are not working (broken).

Since you need to verify the working of every link, the workaround is that, you need to send http requests to all of the links on the web page and analyze the response. Whenever you use driver.get() method to navigate to a URL, it will respond with a status of **200 – OK**. 200 – OK denotes that the link is working and it has been obtained. If any other status is obtained, then it is an indication that the link is broken.

But how will you do that?  
First, we have to use the anchor tags <a> to determine the different hyperlinks on the web page. For each <a> tag, we can use the attribute ‘href’ value to obtain the hyperlinks and then analyze the response received for each hyperlink when used in **driver.get()** method.

**27. Which technique should you consider using throughout the script “if there is neither frame id nor frame name”?**

If neither frame name nor frame id is available, then we can use **frame by index**.

Let’s say, that there are 3 frames in a web page and if none of them have frame name and frame id, then we can still select those frames by using frame (zero-based) index attribute. Each frame will have an index number. The first frame would be at index “0”, the second at index “1” and the third at index “2”. Once the frame has been selected, all subsequent calls on the WebDriver interface will be made to that frame.

|  |  |
| --- | --- |
| 1 | driver.switchTo().frame(int arg0); |

**C. TestNG Framework For Selenium – Selenium Interview Questions**

**28. What is the significance of testng.xml?**

I’m pretty sure you all know the importance of TestNG. Since Selenium does not support report generation and test case management, we use TestNG framework with Selenium. TestNG is much more advanced than JUnit, and it makes implementing annotations easy. That is the reason TestNG framewrok is used with Selenium WebDriver.

But have you wondered where to define the test suites and grouping of test classes in TestNG?

It is by taking instructions from the testng.xml file. We cannot define a test suite in testing source code, instead it is represented in an XML file, because suite is the feature of execution. The test suite, that I am talking about is basically a collection of test cases.

So for executing the test cases in a suite, i.e a group of test cases, you have to create a testng.xml file which contains the name of all the classes and methods that you want to execute as a part of that execution flow.

Other advantages of using testng.xml file are:

* It allows execution of multiple test cases from multiple classes
* It allows parallel execution
* It allows execution of test cases in groups, where a single test can belong to multiple groups

**29. What is parameterization in TestNG? How to pass parameters using testng.xml?**

Parameterization is the technique of defining values in testng.xml file and sending them as parameters to the test class. This technique is especially useful when we need to pass multiple login credentials of various test environments. Take a look at the code below, in which “myName” is annotated as a parameter.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class ParameterizedTest1{   @Test   @Parameters("myName")   public void parameterTest(String myName) {   System.out.println("Parameterized value is : " + myName);   }  } |

To pass parameters using testng.xml file, we need to use ‘parameters’ tag. Look at the below code for example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "<http://testng.org/testng-1.0.dtd>" >   <suite name=”CustomSuite">    <test name=”CustomTest”>     <parameter name="myName" value=”John"/>      <classes>       <class name="ParameterizedTest1" />      </classes>    </test>   </suite> |

To extensively understand the working of TestNG and it’s benefit when used with Selenium, watch the below Selenium tutorial video.

**Selenium Training | TestNG Framework For Selenium | Edureka**

**30. Explain DataProviders in TestNG using an example. Can I call a single data provider method for multiple functions and classes?**

DataProvider is a TestNG feature, which enables us to write DataDriven tests. When we say, it supports DataDriven testing, then it becomes obvious that the same test method can run multiple times with different data-sets. DataProvider is in fact another way of passing parameters to the test method.

**@DataProvider** marks a method as supplying data for a test method. The annotated method must return an Object[] where each Object[] can be assigned to parameter list of the test method.

To use the DataProvider feature in your tests, you have to declare a method annotated by **@DataProvider** and then use the said method in the test method using the ‘dataProvider‘ attribute in the Test annotation.

As far as the second part of the question is concerned, Yes, the same DataProvider can be used in multiple functions and classes by declaring DataProvider in separate class and then reusing it in multiple classes.

**31. How to skip a method or a code block in TestNG?**

If you want to skip a particular test method, then you can set the ‘enabled’ parameter in test annotation to false.  
@Test(enabled = false)

By default, the value of ‘enabled’ parameter will be true. Hence it is not necessary to define the annotation as true while defining it.

**32. What is soft assertion in Selenium? How can you mark a test case as failed by using soft assertion?**

Soft Assertions are customized error handlers provided by TestNG. Soft Assertions do not throw exceptions when assertion fails, and they simply continue to the next test step. They are commonly used when we want to perform multiple assertions.

To mark a test as failed with soft assertions, call **assertAll()** method at the end of the test.

**33. Explain what is Group Test in TestNG?**

In TestNG, methods can be categorized into groups. When a particular group is being executed, all the methods in that group will be executed.  We can execute a group by parameterizing it’s name in group attribute of **@Test** annotation. Example: @Test(groups={“xxx”})

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | @Test(groups={“Car”})  public void drive(){  system.out.println(“Driving the vehicle”);  }    @Test(groups={“Car”})  public void changeGear() {  system.out.println("Change Gears”);  }    @Test(groups={“Car”})  public void accelerate(){  system.out.println(“Accelerating”);  } |

**34. How does TestNG allow you to state dependencies? Explain it with an example.**

**Dependency**is a feature in TestNG that allows a test method to depend on a single or a group of test methods. Method dependency only works if the “depend-on-method” is part of the same class or any of the inherited base classes (i.e. while extending a class). Syntax:  
**@Test(dependsOnMethods = { “initEnvironmentTest” })**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | @Test(groups={“Car”})  public void drive(){  system.out.println(“Driving the vehicle”);  }    @Test(dependsOnMethods={“drive”},groups={cars})  public void changeGear() {  system.out.println("Change Gears”);  }    @Test(dependsOnMethods={“changeGear”},groups={“Car”})  public void accelerate(){  system.out.println(“Accelerating”);  } |

**35. Explain what does @Test(invocationCount=?) and @Test(threadPoolSize=?) indicate.**

**@Test(invocationCount=?)** is a parameter that indicates the number of times this method should be invoked.  
**@Test(threadPoolSize=?)** is used for executing suites in parallel. Each suite can be run in a separate thread.

To specify how many times @Test method should be invoked from different threads, you can use the attribute **threadPoolSize** along with **invocationCount**. Example:

|  |  |
| --- | --- |
| 1  2  3 | @Test(threadPoolSize = 3, invocationCount = 10)  public void testServer() {  } |

So this brings us to the end of the Selenium interview questions blog. Alternatively, you can check out this video on Selenium interview questions delivered by an industry expert where he also talks about the industry scenario and how questions will be twisted and asked in an interview.