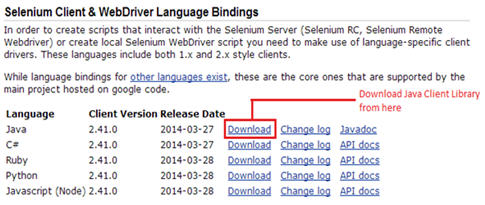
**1)What are limitations of selenium webdriver**

* Selenium only supports web based application and does not support windows based application.
* It is difficult to test Image based application.
* Selenium need outside support for report generation activity like dependence on TestNG Jenkins.
* Selenium does not support built in add-ins support.
* As Selenium is a free tool, thus there is no ready vendor support though the user can find numerous helping communities.
* Selenium does not provide any built in IDE for script generation and it need other IDE like Eclipse for writing scripts.
* Selenium script creation time is bit high.
* Selenium does not support file upload facility.
* Selenium partially supports for Dialog boxes

**2)Installation and configuration of selenium?**

Download the Selenium Java Client Libraries

Step 1: Go to Selenium’s official website and navigate to its download page – “<http://docs.seleniumhq.org/download/>”. Refer the section in the below illustration where you can find Client Libraries listed for distinct programming languages. Click on the download link for Java Client Library.



It may take a few minutes before you can download the complete zipped folder.

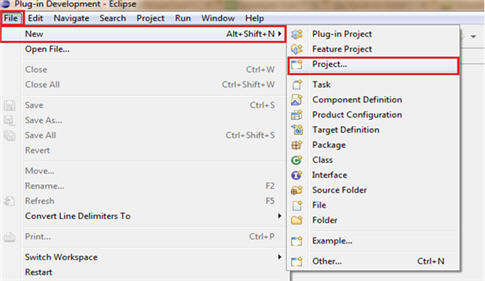
Step 2: Once downloaded, copy the folder and place it in the desired location on your file system.

Step 3: Extract the zipped folder, a folder named as “Selenium-2.41.0.zip“can be seen. The folder embodies all the required jar files which enable users to create test scripts in Java.

Thus these libraries can be configured in Eclipse IDE.

Configuring Libraries with Eclipse IDE

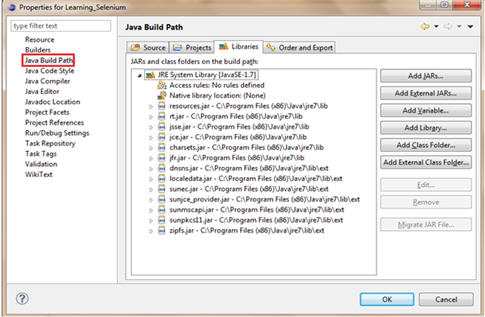
Step 1: Navigate towards Eclipse IDE. Create a new java based project following File -> New -> Java Project. Refer the following figure for the same.



Step 2: Provide a user defined name for your Java Project. Let us provide the name as Learning\_Selenium and Click on the Finish Button. The newly created project can be viewed at the left side of the screen in the package explorer panel.

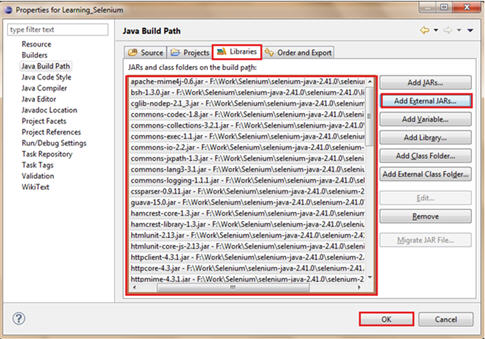
Step 3: Create a new Java class named as “First\_WebdriverClass” under the source folder by right clicking on it and navigating to New -> class.

Step 4: Now let us configure the libraries into our Java project. For this, select the project and Right click on it. Select “Properties” within the listed options. The following screen appears, Select “Java Build Path” from the options.



Step 5: By default, “Libraries” tab is opened. If not, click on the “Libraries” tab. Then, click on the “Add External Jars…” button. Browse to the location where we have saved the extracted folder for Java Client Libraries.

Step 6: Select all the JAR files present in the “selenium-java-2.41.0” folder and click on open button within the dialog box. The properties dialog box should look like the below illustration.



Step 7: Click on the “OK” button within the dialog box so as to complete the configuration part of Selenium Libraries in our java project.

The project will look like the following:



**3) What are different ways of locating elements in selenium?**

1.By ID:

driver.findElement(By.id(“element id”));

2.By CLASS:

Driver.findElement(By.Classname(“element name”);

3.By Name:

driver.findElement(By.name("element name"))

4.By TagName:

Driver.findElement(By.tagName(“element html tag name”));

5.By CSS Selector:

Driver.findElement(By.cssSelector(“css selector”));

6. By Link:

Driver.findElement(By.link(“link text”));

7.By Xpath

Driver.findElement(By.xpath(“xpath expression”);

**4)which is the fastest way to locating elements in selenium?**

The most efficient way and preferred way to locate an element on a web page is By ID. ID will be the unique on web page which can be easily identified.IDs are the safest and fastest locator option and should always be the first choice even when there are multiple choices , It is like an Employee Number or Account which will be unique.

**5) what is absolute path and relative path in xpath.?**

**Absolute Xpath**: It uses Complete path from the Root Element to the desire element.

**Relative Xpath**: You can simply start by referencing the element you want and go from there.

Always Relative Xpaths are preferred as they are not the complete paths from the Root element. (//html//body) ..Beacuse in future any of the webelement when added/Removed then Absolute Xpath changes. So Always use Relative Xpaths in your Automation.

Example:

// Absolute Path starts from root path

WebElement link1 = driver.findElement(By.xpath("/html/body/li[@id='test']/a"));

// Relative Path starts from current path

WebElement link2 = driver.findElement(By.xpath(".//\*[@id='test']/a"));

**6) Different types of waits or synchronizations in selenium webdriver?**

**Synchronization:**

It is a mechanism which involves more than one components to work parallel with Each other.

Generally in Test Automation, we have two components  
**1. Application Under Test**  
**2. Test Automation Tool.**

Both these components will have their own speed. We should write our scripts in such a way that both the components should move with same and desired speed, so that we will not encounter "Element Not Found" errors which will consume time again in debugging.

Synchronization can be classified into two categories:

**1. Unconditional**  
**2. Conditional Synchronization**

**Unconditional :**  
In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.

*Examples: Wait() and [Thread.Sleep();](https://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html" \t "_blank)*

The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.

The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

**Conditional Synchronization:**

We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.

It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.

There are 3 types of wait commands that can be used in Selenium.

1. Implicit Wait
2. Explicit Wait
3. Fluent Wait

**1.Implicit Wait:** Implicit wait can accept the time amount in seconds,milliseconds, nanoseconds,minute etc. It depends on your requirement. Implicit wait sets a default time for selenium WebDriver. An exception will be thrown if it is not able to find out any element within the specified amount of time. This is applicable for each element in the application.

EX:

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://www.gmail.com");

**2. Explicit Wait:** This wait gives you the options to wait until a desired condition is satisfied. You can use some of prebuilt **ExpectedConditions**for the elements to be visible, invisible,clickable etc.

### EX:

### driver = new FirefoxDriver();

### WebDriverWait wait = new WebDriverWait(driver, 10); //Explicit wait for 10 seconds

### driver.get("http://www.gmail.com");

### 

### WebElement username = driver.findElement(By.id("Email");

### wait.until(ExpectedConditions.visibilityOf(username)); //Wait until username is visible

### 3. Fluent Wait:

**FluentWait**contains 3 parameters.

* **WithTimeOut:** defines the maximum amount of time it will wait for any element to be visible.
* **pollingEvery’ :**takes time as an argument e.g. 5 seconds. It will check the status of element after every 5 seconds.
* **ignoring**: we can define the type of exceptions need to be ignored during element search.

**EX:** Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(30, TimeUnit.SECONDS) // wait maximum 30 seconds

.pollingEvery(5, TimeUnit.SECONDS) // check after every 5 seconds

.ignoring(NoSuchElementException.class); // ignore NoSuchElementException

WebElement foo = wait.until(new Function<WebDriver, WebElement>() {

public WebElement apply(WebDriver driver) {

return driver.findElement(By.id("Email"));

}});

**When to use implicit wait, explicit wait and Fluent wait?**

**Implicit wait:**During implicit wait time, it tries to locate the element in the WebPage. If webDriver does not find the element initially. Then it doesn’t try to identify the element again, instead it waits till the specified time is over. In the end moment when time is over, it checks once again if element is present before throwing the exception.The default setting is zero. It is default time which is applied throughout the execution session and only gets destroyed when the WebDriver instance is destroyed.

**Explicit Wait:**It should be applied for those specific elements which takes more time to be displayed comparing to other elements in the application. Suppose, we have observed that one element is taking around 1 minute time to be displayed. Then we can not apply implicit wait for 1 minute because then it will wait for same amount of time for every element.

Instead, we will use explicit wait with the expected condition.

**Fluent Wait:**Suppose, there are few elements which sometimes takes more than 1 minute to display or sometimes gets displayed within 2 seconds. For such elements, it would be better to use fluent wait. As it will keep checking the presence of those elements  after a specified amount of time until it finds it or time out.

FluentWait can prove pretty handy for Ajax because of the flexibility you have for ignoring exceptions and easily tweaking the wait and polling times.

7) How to save screen shots using selenium webdriver?

Its very important to take screenshot when we execute a test script. When we execute huge number of test scripts, and if some test fails, we need to check why the test has failed.

It helps us to debug and identify the problem by seeing the screen shot.

In selenium webdriver, we can take the screen shot using the below command.  
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

8) How to handle multiple windows in selenium webdriver?

**Selenium WebDriver** software testing tool has built in "**WebDriver.switchTo().window()**" method available to switch from one window to another window so it is very easy to **handle multiple windows in webdriver**.

**WebDriver.getWindowHandles()**  
In WebDriver software testing tool, We can use "WebDriver.getWindowHandles()" to get the handles of all opened windows by webdriver and then we can use that window handle to switch from from one window to another window. Example Syntax for getting window handles is as bellow.

Set<String> AllWindowHandles = driver.getWindowHandles();

**WebDriver.switchTo().window()**  
WebDriver.switchTo().window() method is useful to switch from one window to another window of software web application. Example syntax is as bellow.

driver.switchTo().window(window2);

Ex:

Before commencing, let us first consider a few situations when we are likely to deal with multiple windows.

* Filling forms may require to select the date from a separately opened window.
* Clicking on some link/button can kick-off yet another window.
* Handling Advertisement windows

Hence, we can come up with various scenarios depending upon the application.  
Now let us motion ourselves towards the challenge we face under above situations. The most particular of all is switching the focus from one window to another. Let us understand the same in the following way:

Comprehending from the above figure, the entire process can be fundamentally segregated into following steps:

**Step 1 :** Clicking on Link1 on Window A

A new Window B is opened.

**Step 2 :** Move Focus from Window A to Window B

Window B is active now

**Step 3 :** Perform Actions on Window B

Complete the entire set of Actions

**Step 4 :** Move Focus from Window B to Window A

Window A is active now

**These are the steps** which we can easily interpret out of the diagram, but there are a few more steps to add to complete this process and making our script execute. These steps don’t have visibility but plays a very vital role. Let us now re-consider the same scenario.

**Step 1 :** Clicking on Link1 on Window A

**A new Window B is opened.**

Step 2 :**Save reference for Window A**

Step 3 :**Create reference for Window B**

**Step 3 :** Move Focus from Window A to Window B

**Window B is active now**

**Step 3 :** Perform Actions on Window B

Complete the entire set of Actions

**Step 4 :** Move Focus from Window B to Window A

**Window A is active now**

9) How to launch webpage using chrome driver?

Using system .set property we need specify the exe file path of chrome, Then we can use webdriver to open the chrome window.

EX: System.*setProperty*("webdriver.chrome.driver", "File location");

WebDriver driver=**new** ChromeDriver();

10) what is desired capabilities in selenium webdriver?

The desired capability is a series of key/value pairs that stores the browser properties like browsername, browser version, the path of the browser driver in the system, etc. to determine the behaviour of the browser at run time.

Desired capability can also be used to configure the driver instance of Selenium WebDriver. We can configure driver instance like FirefoxDriver, ChromeDriver, InternetExplorerDriver by using desired capabilities. Desired Capabilities are more useful in cases like:

In mobile application automation, where the browser properties and the device properties can be set. In Selenium grid when we want to run the test cases on a different browser with different operating systems and versions.

1. Mostly DesiredCapabilities class used when do we used Selenium Grid.
2. We have to execute mutiple TestCases on multiple Systems with different browser with Different version and Different Operating System.

11) how to set language while opening website?

**Internationalization** is a process of designing a software application so that it can be adapted to various languages and regions without any changes

**Localization** is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

If we want to check whether our application is properly internationalized , then we will manually change the language preferences in the browser itself.But if we want to check the same using WebDriver then we have to change the user language preferences.

**Using Firefox Browser :**

FirefoxProfile profile = new FirefoxProfile();  
//setting the locale french : ‘fr’  
profile.setPreference(“intl.accept\_languages”,”fr”);  
driver = new FirefoxDriver(profile);  
driver.get(“[http://google.co.in&#8221](#8221););  
**Using Chrome Browser :**

System.setProperty(“webdriver.chrome.driver”,”D:/DollarArchive/chromedriver.exe”);  
ChromeOptions options = new ChromeOptions();  
options.addArguments(“–lang= sl”);  
ChromeDriver driver = new ChromeDriver(options);  
driver.get(“[http://google.co.in&#8221](#8221););

Unfortunately it wont work for IE browser, We need to change it manually.

12) how to handle windows based popups (upload and dropdown).

**Handling web based pop-up box**

WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.

**There are the four methods that we would be using along with the Alert interface.**

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

13) write code to verify any application login page is working or not?

package Basics;

import static org.junit.Assert.assertTrue;

import java.net.MalformedURLException;

import java.net.UnknownHostException;

import org.junit.Before;

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

/\*13) write code to verify any application login page is working or not?\*/

public class LoginPageTest{

//private HtmlUnitDriver driver;

WebDriver driver;

@Before

public void setup() throws MalformedURLException, UnknownHostException, InterruptedException{

driver = new FirefoxDriver();

Thread.sleep(5000);

driver.get("https://www.facebook.com/");

}

@Test

public void testAuthenticationFailureWhenProvidingBadCredentials(){

driver.findElement(By.id("email")).sendKeys("fakeuser");

driver.findElement(By.id("pass")).sendKeys("fakepassword");

driver.findElement(By.id("u\_0\_o")).click();

assertTrue(driver.getCurrentUrl().endsWith("failed"));

}

@Test

public void testAuthenticationSuccessWhenProvidingCorrectCredentials(){

driver.findElement(By.id("email")).sendKeys("akhilashlakma@gmail.com");

driver.findElement(By.id("pass")).sendKeys("akhil534");

driver.findElement(By.id("u\_0\_o")).click();

assertTrue(driver.getCurrentUrl().endsWith("/<facebook>/"));

}

}

14) how to select items from dropdown/select box.

If we want to work with select and drop down we need to ***import org.openqa.selenium.support.ui.Select***' package and then we can create a object for select class. Select is a class which is provided by Selenium to perform multiple operations on DropDown object and Multiple Select object. This class can be found under the ***Selenium’s Support.UI.Select*** package. As Select is also an ordinary class, so it’s object is also created by a New keyword with regular class creation syntax.

**Select oSelect =newSelect(**element**));**

Ex: WebElement element = driver.findElement(By.id("Country"));

Select oSelect = new Select(element);

//Or it can be also written as

Select oSelect = new Select(driver.findElement(By.id("Country")));

15) how to know if checkbox is checked or not in webpage.

## By ID

If **ID**is given for the Radio Button/CheckBox and you just want to click on it irrespective of it’s value, then the command will be like this:

EX:

WebElement radioBtn = driver.findElement(By.id("toolsqa"));

radioBtn.click();

## With IsSelected

If your choice is based on the pre-selection of the Radio Button/Check Box and you just need to select the deselected Radio Button/Check Box. Assume there are two Radio Buttons/Check Boxes, one is selected by default and you want to select the other one for your test. With ***IsSelected* statement, you can get to know that the element is selected or not.**

## With Value

You can even select Radio Buttons/Check Boxes with their Values.

## By CssSelector

A simple way of selecting a check-box or radio button is by using its value:

EX:

WebElement oCheckBox = driver.findElement(By.cssSelector("input[value='Tools QA']"));

oCheckBox.click();

16) tell me code to pass values from parent window to child window?

17) write code to find out if all links are working or not?

Finding broken links on a web page can be classified into two steps

1. Finding all the links on the page.

2. Iteratively checking the links if they are broken.

For checking the links we will take the help of java class called HttpURLConnection class. This class is used to  make HTTP requests to the webserver hosting the links extracted in step 1. and see the response returned by the server. Based on the response we can figure out if the link is broken or not.

18)write code on how to use javascriptexecutor?

[JavaScriptExecutor](http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/JavascriptExecutor.html) is an interface which provides mechanism to execute Javascript through selenium driver. It provides “executescript” & "executeAsyncScript" methods, to run JavaScript in the context of the currently selected frame or window.

Package:-  
  
import org.openqa.selenium.JavascriptExecutor;  
  
Syntax:-

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(Script,Arguments);

*script* - The JavaScript to execute

*Arguments* - The arguments to the script.(Optional)

19) difference between assert and verify?

Both Assert and Verify commands are used to find whether a given input is present or not on the webpage. There are some difference between Assert and Verify in Selenium.

Let’s see the basic difference between Assert and Verify in Selenium:

**Assert command in selenium:**

When an “assert” command fails, the test execution will be aborted. So when the Assertion fails, all the test steps after that line of code are skipped. The solution to overcoming this issue is to use a try-catch block. We use the Assertion in the try catch block. Mostly, the assert command is used when the end result of the check value should pass to continue to the next step.

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 command in selenium:**

When a “verify” command fails, the test will continue executing and logging the failure. Mostly, the Verify command is used to check non-critical things. In such cases where we move forward even though the end result of the check value is failed.

In simple words, there wont be any halt in the test execution even though the verify condition is true or false.

**Note:** In TestNG, we use only Assert Statements.

20) difference between driver.close and driver.quit methods?

driver.close and driver.quit are two different methods for closing the browser session in Selenium WebDriver. Understanding both of them and knowing when to use which method is important in your test execution. Therefore, in this article, we have tried to throw light on both these methods.

* driver.close – It closes the the browser window on which the focus is set.
* driver.quit – It basically calls driver.dispose method which in turn closes all the browser windows and ends the WebDriver session gracefully.

You should use driver.quit whenever you want to end the program. It will close all opened browser window and terminates the WebDriver session. If you do not use driver.quit at the end of program, WebDriver session will not close properly and files would not be cleared off memory. This may result in memory leak errors.

EX: It depends on what you are trying to achieve. As others have pointed out,  
using close() will close the current window and quit() will quit the  
browser. If I was testing a website manually, would I close the window but  
leave the browser running? This will probably affect things like cookies,  
saved data, history, etc. If the requirement is that I can log in, it  
creates a permanent cookie, I close the browser, I open the browser, it  
remembers my username then I'm going to have to quit() the browser in  
Selenium.  
  
If it does not make a difference and I always leave the browser open and  
just create new windows then I might use an @BeforeClass to open the  
browser once, each test will open a new window (so if there are 30 tests in  
the suite it will end with 30 windows open) then at the end of all the  
testing I'll use an @AfterClass to quit() and close all the windows.

21) common exceptions in selenium?

There is a complete list of Exceptions mentioned on the Selenium Doc which you may or may not encounter in course of your  testing.

**Most common Exceptions:**

1) NoSuchElementException : FindBy method can’t find the element.

2) StaleElementReferenceException : This tells that element is no longer appearing on the DOM page.

3) TimeoutException: This tells that the execution is failed because the command did not complete in enough time.

4) ElementNotVisibleException: Thrown to indicate that although an element is present on the DOM, it is not visible, and so is not able to be interacted with

5) ElementNotSelectableException: Thrown to indicate that may be the element is disabled, and so is not able to select.

22) how to handle Ajax calls in selenium?

The **biggest challenge in handling Ajax call is knowing the loading time for the web page.**Since the loading of the web page will last only for a fraction of seconds, it is difficult for the tester to test such application through automation tool. For that, Selenium Webdriver has to use the wait method on this Ajax Call.

So by executing this wait command, selenium will suspend the execution of current test case and wait for the expected or new value. When the new value or field appears, the suspended test cases will get executed by Selenium Webdriver.

Following are the wait methods that Selenium Webdriver can use

1. **Thread.Sleep()**

* Thread.Sleep () is not a wise choice as it suspends the current thread for the specified amount of time.
* In AJAX, you can never be sure about the exact wait time. So, your test will fail if the element won't show up within the wait time. Moreover, it increases the overhead because calling Thread.sleep(t) makes the current thread to be moved from the running queue to the waiting queue.
* After the time 't' reached, the current thread will move from the waiting queue to the ready queue, and then it takes some time to be picked by the CPU and be running.

1. **Implicit Wait()**

* This method tells webdriver to wait if the element is not available immediately, but this wait will be in place for the entire time the browser is open. So any search for the elements on the page could take the time the implicit wait is set for.

1. **Explicit Wait()**

* [Explicit wait](http://www.guru99.com/implicit-explicit-waits-selenium.html) is used to freeze the test execution till the time a particular condition is met or maximum time lapses.

1. **WebdriverWait**

* It can be used for any conditions. This can be achieved with WebDriverWait in combination with ExpectedCondition
* The best way to wait for an element dynamically is checking for the condition every second and continuing to the next command in the script as soon as the condition is met.

But the problem with all these waits is, you have to mention the time out unit. What if the element is still not present within the time? So there is one more wait called Fluent wait.

1. **Fluent Wait**

* This is an implementation of the Wait interface having its timeout and polling interval. Each FluentWait instance determines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition.

### Challenges in Handling Ajax Call in Selenium Webdriver

* Using "pause" command for handling Ajax call is not completely reliable. Long pause time makes the test unacceptably slow and increases the testing time. Instead, "waitforcondition" will be more helpful in testing Ajax applications.
* It is difficult to assess the risk associated with particular Ajax applications
* Given full freedom to developers to modify Ajax application makes the testing process challenging
* Creating automated test request may be difficult for testing tools as such AJAX application often use different encoding or serialization technique to submit POST data.

**Summary:**

* AJAX allows the Web page to retrieve small amounts of data from the server without reloading the entire page.
* To test Ajax application, different wait methods should be applied
  + ThreadSleep
  + Implicit Wait
  + Explicit Wait
  + WebdriverWait
  + Fluent Wait
* Creating automated test request may be difficult for testing tools as such AJAX application often use different encoding or serialization technique to submit POST data.

EX:

### AJAX Call Test Case Scenario.

#### Demo Site Used:

We’ve used the following demo URL for our testing which is using the AJAX calls.

[**http://www.w3schools.com/ajax/tryit.asp?filename=tryajax\_callback**](http://www.w3schools.com/ajax/tryit.asp?filename=tryajax_callback)

#### Test Case Description:

1. Open the demo AJAX application demo website.
2. Following AJAX controls would appear in an IFRAME.
   1. A demo paragraph element which contains some default text.
   2. A simple button control to make the AJAX calls.
3. When you click the button control, the AJAX call takes place.
   1. The default text disappears from the screen.
   2. Two new paragraphs get displayed on the screen.
4. You now need to validate the two conditions.
   1. The new text in the first paragraph shouldn’t match the default text.
   2. The text in the second paragraph should match the expected value.

***Note:***We’ll read the default text from the demo paragraph as it would appear first on the screen. For the second paragraph, we’ve hard wired the value in the sample code which we copied from the demo test site.

23) we have webtable, need to click on second row from table,when we click on child will be populated. first column in primary column for each row. tell me steps to verify child form has proper data or not

24)How to assign the value to textbox other than sendkeys method?

We can type text In text box of software web application page using bellow given ways In selenium test.  
  
**1.Using .SendKeys() method**

driver.findElement(By.xpath("//input[@id='fname']")).sendKeys("Using sendKeys");

**2. Using JavascriptExecutor**

((JavascriptExecutor)driver).executeScript("document.getElementById('fname').value='Using JavascriptExecutor'");

25) Write code for drag/drop in selenium.

For all advance activity in Selenium Webdriver, we can perform easily using Actions class like Drag and Drop, mouse hover, right click, Click and Hold and so on.  
We have predefined method called **dragAndDrop(source, destination)** which is a method of Actions class.

Approach- Find the xpath of the Source  and find the xpath of destination.

Both source and destination in form of WebElement.

Note- Any method of Actions class we need to call perform () method otherwise we will get anexception. If we have series of action in our script using Actions class then we have to call build().perform() method.

26) Write code for right click in selenium.

create an instance of Action class and call the methods defined in it to perform the actions you want. Follow the below steps in order to perform right click on any web element :

1) Launch Eclipse, create a project and a package under it, and then a class named “tstclss.java”.

2) Write the code to navigate to URL “http://seleniumtutorialpoint.com/”.

3) There is an option “SAMPLE PAGE” available on the upper left side of the side. We have to right click on it and have to open it in a new tab.

The line of code you need to write in order to navigate to this site is :

**WebDriver driver=new FirefoxDriver();**  
**driver.get(“http://seleniumtutorialpoint.com/”);**

4) Now we have to perform right click on the option “Sample Page”. In order to do it, first we will create an instance of Action class and we will pass driver instance into it. But before creating the instance, we need to locate the web element “Sample Page” which is :

**WebElement ele=driver.findElement(By.xpath(“//\*[@id=’navigation’]/ul/li/a”));**

The line of code which is to be added in the script for creation of Action class and performing methods is :

**Actions act=new Actions(driver);**  
**act.contextClick(ele).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.ENTER).build().perform();**

Here the method sendKeys(Keys.ARROW\_DOWN) is used to select an option from the list. If you will not add this method, the right click on the web element will be performed and the option list which appears after the right click will get disappeared without selecting any option.

We can perform the right click on any web element using the Robot class too. Here you first need to move the mouse to the particular web element and then perform the required action…!!!

27) Write code for scroll to specific element.

**Scroll by given pixel offset**

I have created test script to scroll down-up web page In horizontal or vertical direction based on given x y pixel offset.

Bellow given example will first scroll down(X direction) web page by 600 pixels and then It will scroll up It by 300 pixels.

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Scrolling {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver =new FirefoxDriver();

driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://only-testing-blog.blogspot.in");

}

@Test

public void Scroll\_Page() throws IOException, InterruptedException {

**//To scroll down web page by 600 pixels In x(vertical) direction.**

**//You can y parameter to scroll page In horizontal direction.**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

javascript.executeScript("window.scrollBy(0,600)", "");

Thread.sleep(3000);

**//To scroll up web page by 300 pixels In x(vertical) direction.**

javascript.executeScript("window.scrollBy(0,-300)", "");

}

}

**Scroll down to bottom of page**

If you wants to scroll down to bottom of the page, You can use bellow given @Test method In above example.

@Test

public void Scroll\_Page() throws IOException, InterruptedException {

**//Scroll down to bottom of the page.**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

javascript.executeScript("window.scrollTo(0, document.body.scrollHeight)", "");

}

**Scroll to element In selenium WebDriver**  
If you wants to scroll till some element on page then you can use element parameter In javascript as shown In bellow example.

**//Scroll till element.**

JavascriptExecutor je = (JavascriptExecutor) driver;

WebElement element = driver.findElement(By.xpath("//div[@id='dragdiv']"));

je.executeScript("arguments[0].scrollIntoView(true);",element);

28) Selenium grid, how to execute scripts on multiple browser?

**Selenium Grid is a part of the Selenium Suite that specializes on running multiple tests across different browsers, operating systems, and machines in parallel**.

Selenium Grid uses a hub-node concept where you only run the test on a single machine called a **hub**, but the execution will be done by different machines called **nodes**.

You should use Selenium Grid when you want to do either one or both of following:

* **Run your tests against different browsers, operating systems, and machines all at the same time.**This will ensure that the application you are testing is fully compatible with a wide range of browser-O.S combinations.
* **Save time in the execution of your test suites**. If you set up Selenium Grid to run, say, 4 tests at a time, then you would be able to finish the whole suite around 4 times faster.

**The Hub**

* The hub is the central point where you load your tests into.
* There should only be one hub in a grid.
* The hub is launched only on a single machine, say, a computer whose O.S is Windows 7 and whose browser is IE.
* The machine containing the hub is where the tests will be run, but you will see the browser being automated on the node.

**The Nodes**

* Nodes are the Selenium instances that will execute the tests that you loaded on the hub.
* There can be one or more nodes in a grid.
* Nodes can be launched on multiple machines with different platforms and browsers.
* The machines running the nodes need not be the same platform as that of the hub.