SELENIUM TOOLS

Introduction to selenium tools(Selenium IDE, Core, RC)

Selenium is an open source tool for web application testing. Selenium tests run directly in a browser, just like real users do. It runs in Internet Explorer, Mozilla Firefox on Windows, Linux, and Macintosh, Safari on the Mac.

There are three variants of Selenium, which can be used in isolation or in combination to create complete automation suite for your web applications.

- **Selenium IDE :**

Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, debug and execute tests. It can also record user actions in most of the popular languages like Java, C#, Perl, Ruby etc. This eliminates the need of learning new vendor scripting language. For executing scripts created in these languages, you will need to use Selenium Remote Control. If you do not want to use Remote Control than you will need to create your test script in HTML format.

Biggest drawback of Selenium IDE is its limitation in terms of browser support. Though Selenium scripts can be used for most of the browser and operating system, Scripts written using Selenium IDE can be used for only Firefox browser if it is not used with Selenium RC or Selenium Core.

- **Selenium Core :**

Selenium Core is a test tool for web applications. Selenium Core tests run directly in a browser. And they run in Internet Explorer, Mozilla Firefox on Windows, Linux, and Macintosh. But to use Selenium Core we need to make it available from the same web server as the application you want to test(AUT).

- **Selenium Remote Control :**

Selenium Remote Control (RC) is a test tool that allows you to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser**.**

Selenium RC comes in two parts.

1. A server which can automatically launch and kill supported browsers, and acts as a HTTP proxy for web requests from those browsers.
2. Client libraries for your favorite computer language.

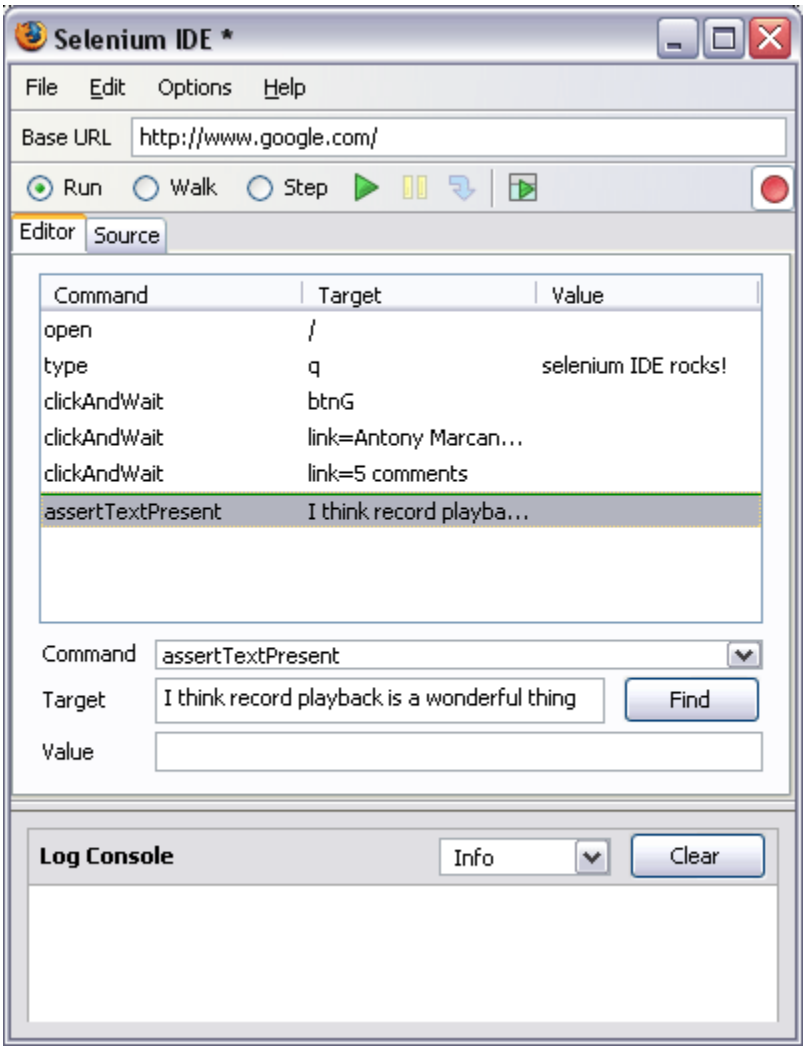
Selenium IDE:

Installation process for Selenium IDE

Open the download link using Mozilla(version: 1.5 or higher) and choose any version of Selenium IDE and select INSTALL NOW. It will get installed automatically. The download link is given below.

Download Link: http://selenium-ide.openqa.org/download.jsp

Working :



Once the installation is successfully completed, go to 'Tools' in the firefox window. You can find Selenium IDE. Click on Selenium IDE. The small red button on the right hand side gives you an indication on whether Selenium is in recording mode or not. Click the red button to start recording and click the red button again to stop the recording.

Run will execute the tests with the maximum possible speed. Walk will execute them with relatively slow speed. In Step mode you will need to tell Selenium to take small steps. Green tilted triangular button is to execute the test. Yellow button to pause while executing the test. Blue button helps to place checkpoints and the final green button is the Selenium Test Runner.

Test Runner gives you nice browser interface to execute your tests and also gives summary of how many tests were executed, how many passed and failed. It also gives similar information on commands which were passed or failed. Test Runner is available to tests developed in HTML only.

In selenium, there is option to start a new test, save test and open the saved test. It is also possible to export scripts. We have other self explanatory options like encoding of test files, timeout under the Options tab. When we access the Format tab under Options tab, we have an option as to specify what kind of formatting we would like in the generated code as Selenium IDE can generate code in variety of languages.

Recording and Executing a Test:

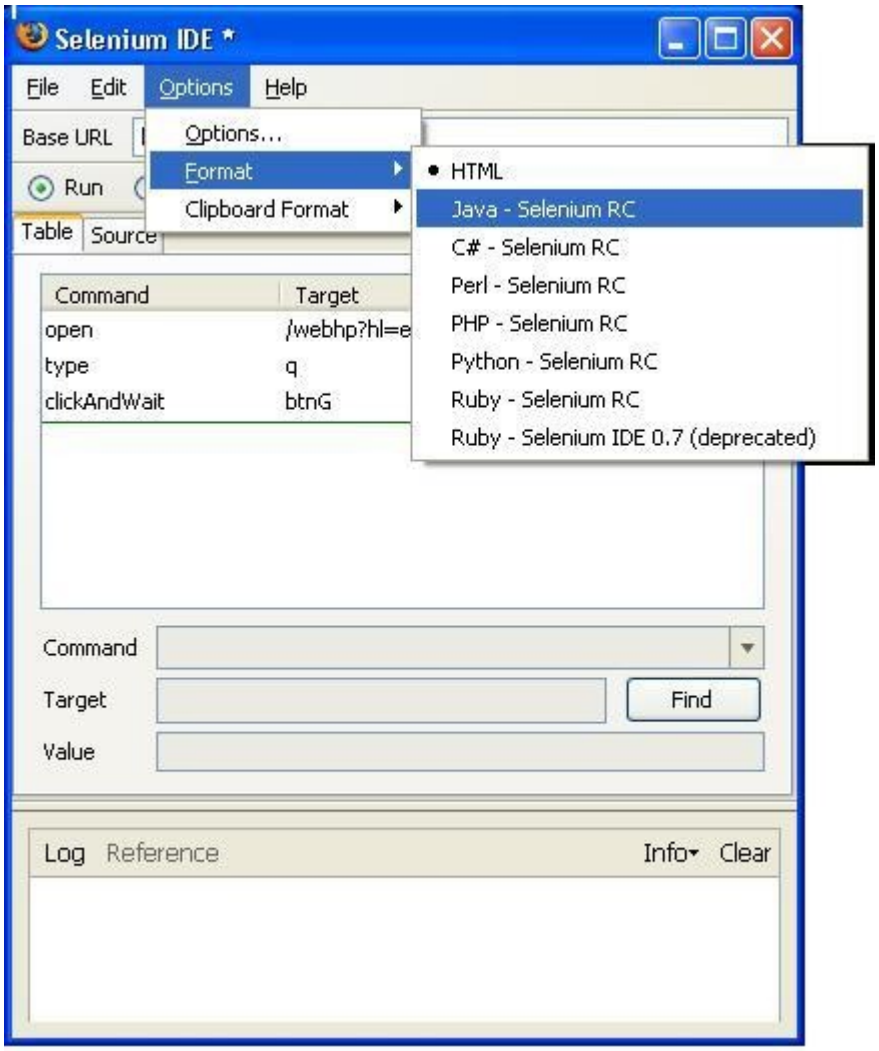
- Make sure you have installed Selenium IDE in Firefox.
- Open Firefox and application you want to test
- Launch Selenium IDE using *tools-Selenium IDE*
- By default, you should be in the recording mode, but confirm it by observing the Red button.
- By default it will be in the HTML format. Otherwise, go to Options-Format-Select HTML Format.
- Record some actions and make sure that these are coming on Selenium IDE.
- During recording if you right click on any element it will show all the selenium commands available.
- You can also edit existing command, by selecting it and editing on the boxes available.
- You can also insert/delete commands by choosing appropriate option after right clicking.
- Choose appropriate run option - i.e walk, run or test runner and review your results.

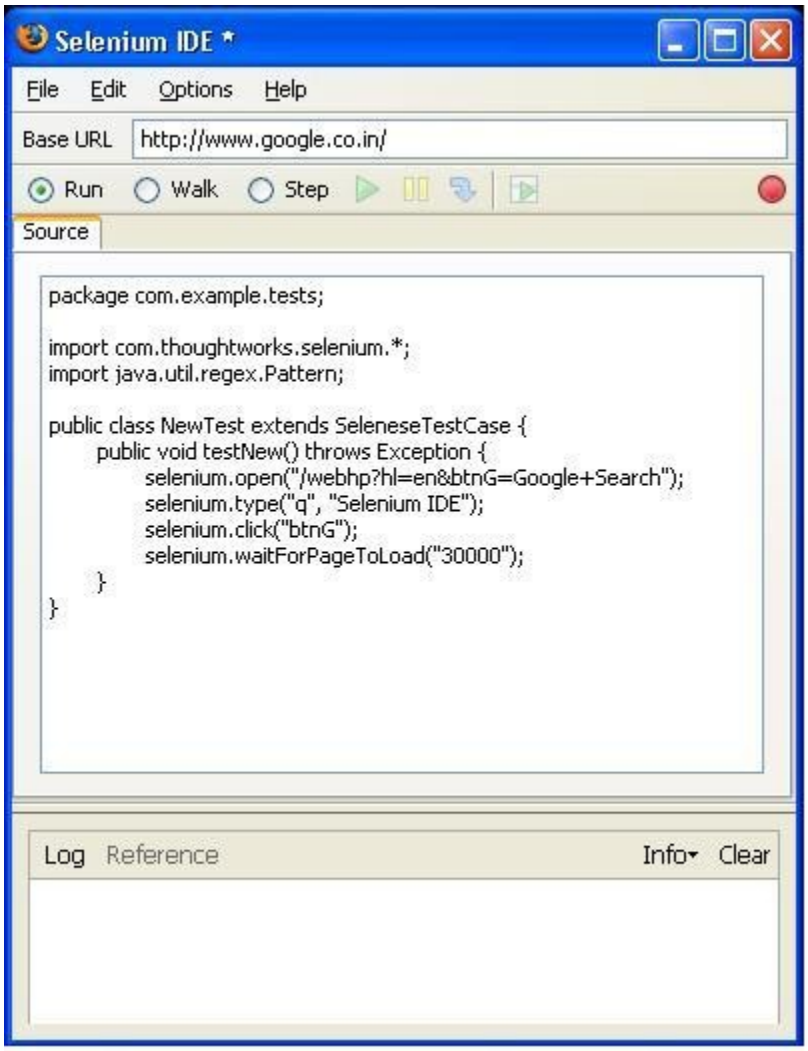
Video presentation to record and execute a test :

http://wiki.openqa.org/download/attachments/400/Selenium+IDE.swf?

Converting the recorded script into different Languages :

- Open Firefox and application you want to test.
- Record operations on the web application which you want to test
- Insert check points
- Go to Options menu – Format – Select any language





Advantages:

- It is a Freeware
- Simple, Easy to install, Easy to work
- Selenium IDE is the only flavor of Selenium which allows you to record user action on browser window
- Can also record user actions in most of the popular languages like Java, C#, Perl, Ruby
- It will not record any operation that you do on your computer apart from the events on Firefox browser window
- During recording if you right click on any element it will show all the selenium commands available
- You can also edit existing command, by selecting it and editing on the boxes available

- You can also insert/delete commands by choosing appropriate option after right clicking
- Choose appropriate run option - i.e walk, run or test runner and review your results

Disadvantages:

- Limitation in terms of browser support (It runs only in Mozilla)
- We can't run recorded script if it is converted to Java, C#, Ruby etc.
- Not allowed to write manual scripts like conditions and Loops for Data Driven Testing
- There is no option to verify images.

Selenium RC:

Installing process for Selenium RC :

The Selenium Server is written in Java, and requires the Java Runtime Environment (JRE) version 1.5.0 or higher in order to start. You may already have it installed. Try running this from the command line:

java -version

You should see a brief message telling you what version of Java is installed, like this:

java version "1.5.0_07"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_07-b03)
Java HotSpot(TM) Client VM (build 1.5.0_07-b03, mixed mode)

If you see an error message instead, you may need to install the JRE, or you may need to add it to your PATH environment variable if it is already installed.

Open the link mentioned below and download Selenium RC. Extract it, you will get Selenium Server folder and Java, Ruby, Perl, Dot Net, PHP, Python Client folders also. Make sure that whether 'selenium-server.jar' file exist in the server folder or not. Open Command window and type the command to start the server.

Java -jar selenium-server.jar (non-interactive mode)

Java -jar selenium-server.jar -interactive (interactive mode)

Interactive mode: In this mode user ca execute commands individually. For example, to open google.com in the firefox, the command to be typed is

cmd=getNewBrowserSession&1=***firefox**&2=**http://www.google.com**

Non-Interactive mode: In this mode server will be ready to take and process HTTP web requests when user running the selenium scripts.

Download ink : http://selenium-rc.openqa.org/download.html

Starting Java Client :

- Download Junit software
- Add the path of junit-4.0.jar file to CLASSPATH
- Add path of selenium-java-client-driver.jar to CALSSPATH
- Write the Selenium Script in Java using any java editor
- Compile and execute it

Note: Start the Selenium Server before executing the script.

Example Selenium Script using Java:

```
import com.thoughtworks.selenium.*;
import junit.framework.*;
import java.util.regex.Pattern;

public class PBNLoginTest extends SeleneseTestCase
{

	private Selenium selenium;

	public void setUp()
	{
		String BaseURL = "http://192.168.1.132/pbn/";
		selenium = new DefaultSelenium("SELENIUM_SERVER_IP", 4444,
"*iexplore", BaseURL);
		selenium.start();
	}
	public void tearDown()
	{
		selenium.stop();
	}
```

```
	public static void main(String[] args)
	{
		PBNLoginTest e = new PBNLoginTest();
		e.setUp();
		e.PBNLogin();
		e.tearDown();
	}


	public void PBNLogin()
	{
		int Total_Check_Points = 1;
		int Passed = 0, Failed = 0;

		selenium.open("/pbn/Default.aspx");
		//Checking Login using valid LoginID and valid password
		selenium.type("txtLogin", "Rajasekhar.Chintha");
		String Login = selenium.getValue("txtLogin");
		selenium.type("txtPassword", "abcd1234");
		selenium.click("link=GO");
		selenium.waitForPageToLoad("30000");
		selenium.selectFrame("link");

		boolean Login_Name = selenium.isTextPresent("User*"+Login+"*");

		if (!Login_Name)
		{
			System.out.println("Failed : Text 'User : " + Login + "(Test
Analyst)' is not found");
			Failed++;
		}
		else
		{
			System.out.println("Passed : Text 'User : " + Login + "(Test
Analyst)' is found");
			Passed++;
		}

		selenium.click("link=Logout");

		System.out.println("Total Check Points : " + Total_Check_Points + "
Passed : " + Passed + " Failed : " + Failed);

	}
```

```
}
```

Note: To compile and execute the script, use the following commands

Javac scriptname.java **(To compile the script)**

Java scriptname **(To execute the script)**

Features :

- We can use Java syntax to write test script
- Easy to conduct Data Driven Testing
- We can read files to get test data
- We can store Test Results into files

Disadvantages :

- There are no Results File generated by Selenium RC. We have to store results in files or etc using Java.

Launching Browsers:

To launch browsers, use ***firefox**, ***chrome**, ***iexplore**, ***iehta** in the **DefaultSelenium** command.

- ***firefox** and ***iexplore** are used to launch FireFox and Internet Explorer for HTTP web pages respectively.
- ***chrome** and ***iehta** are used to launch FireFox and Internet Explorer for HTTPS web pages respectively.

Some Important Server Commands:

Usage: **java -jar selenium-server.jar [-interactive] [options]**

- **-port <nnnn>**: The port number the selenium server should use (default 4444)
- **-timeout <nnnn>**: An integer number of seconds before we should give up

- **-interactive**: Puts you into interactive mode.
- **-multiWindow**: Puts you into a mode where the test web site executes in a separate window, and selenium supports frames
- **-forcedBrowserMode <browser>**: Sets the browser mode (e.g. "*iexplore" for all sessions, no matter what is passed to getNewBrowserSession)
- **-htmlSuite <browser> <startURL> <suiteFile> <resultFile>**: Run a single HTML Selenese (Selenium Core) suite and then exit immediately, using the specified browser on the specified URL. You need to specify the absolute path to the HTML test suite as well as the path to the HTML results file we'll generate

Element Locators:

Element Locators tell Selenium which HTML element a command refers to. The format of a locator is:

locatorType=argument

A locator type can be an element id, an element name, an xpath expression, link text, and more.

Examples:-

```
selenium.click("id=idOfThing");                                  (an id locator)
selenium.click("name=nameOfThing");                              (a name locator)
selenium.click("xpath=//img[@alt='The image alt text']"); (an xpath locator)
selenium.click("dom=document.images[56]" );                      (DOM locator)
selenium.click("link=Test Page For Selenium");                   (a link locator)
selenium.click("css=span#firstChild");                           (a css locator)
```

Frequently used Selenium Commands:

S.No	Command	Description
1	assignId("Locator","String")	Temporarily sets the "id" attribute

		of the specified element
2	captureScreenshot("File name")	Captures a PNG screenshot to the specified file.
3	Check("Locator")	Check a toggle-button (checkbox/radio)
4	click("Locator")	Clicks on a link, button, checkbox or radio button.
5	clickAt("Locator","Coordinate String")	Clicks on a link, button, checkbox or radio button.
6	close()	Simulates the user clicking the "close" button in the title bar of a popup window or tab.
7	doubleClick("Locator")	Double clicks on a link, button, checkbox or radio button.
8	doubleClickAt("Locator","Coordinate String")	Double clicks on a link, button, checkbox or radio button.
9	getAlert()	Retrieves the message of a JavaScript alert generated during the previous action, or fail if there were no alerts.
10	getAllButtons()	Returns the IDs of all buttons on the page.
11	getAllFields()	Returns the IDs of all input fields on the page.
12	getAllLinks()	Returns the IDs of all links on the page.
13	getAllWindowIds()	Returns the IDs of all windows that the browser knows about.
14	getAllWindowNames()	Returns the names of all windows that the browser knows about.
15	getAllWindowTitles()	Returns the titles of all windows that the browser knows about.
16	getAttribute("Attribute Locator")	Gets the value of an element attribute.
17	getBodyText()	Gets the entire text of the page.
18	getConfirmation()	Retrieves the message of a JavaScript confirmation dialog generated during the previous action.
19	getCookie()	Return all cookies of the current page under test.
20	getElementHeight("Locator")	Retrieves the height of an element
21	getElementPositionLeft("Locator")	Retrieves the horizontal position of an element
22	getElementPositionTop("Locator")	Retrieves the vertical position of an element

23	getElementWidth("Locator")	Retrieves the width of an element
24	getEval("JS Expression")	Gets the result of evaluating the specified JavaScript snippet.
25	getLocation()	Gets the absolute URL of the current page.
26	getMouseSpeed()	Returns the number of pixels between "mousemove" events during dragAndDrop commands (default=10).
27	getPrompt()	Retrieves the message of a JavaScript question prompt dialog generated during the previous action.
28	getSelectedId("Select Locator")	Gets option element ID for selected option in the specified select element.
29	getSelectedIds("Select Locator")	Gets all option element IDs for selected options in the specified select or multi-select element.
30	getSelectedIndex("Select Locator")	Gets option index (option number, starting at 0) for selected option in the specified select element.
31	getSelectedIndexes("Select Locator")	Gets all option indexes (option number, starting at 0) for selected options in the specified select or multi-select element.
32	getSelectedLable("Select Locator")	Gets option label (visible text) for selected option in the specified select element.
33	getSelectedLables("Select Locator")	Gets all option labels (visible text) for selected options in the specified select or multi-select element.
34	getSelectedValue("Select Locator")	Gets option value (value attribute) for selected option in the specified select element.
35	getSelectedValues("Select Locator")	Gets all option values (value attributes) for selected options in the specified select or multi-select element.
36	getSelectOptions("Select Locator")	Gets all option labels in the specified select drop-down.
37	getSpeed()	Get execution speed (i.e., get the millisecond length of the delay following each selenium operation).

38	getTable("Table Cell Address”)	Gets the text from a cell of a table.
39	getText("Locator")	Gets the text of an element.
40	getTitle()	Gets the title of the current page.
41	getValue("Locator")	Gets the (whitespace-trimmed) value of an input field (or anything else with a value parameter).
42	getWhetherThisFrameMatchFrameExpression("Current Frame","Target")	Determine whether current/locator identify the frame containing this running code
43	getWhetherThisWindowMatchWindowExpression("Current Window","Target")	Determine whether currentWindow String plus target identify the window containing this running code.
44	goBack()	Simulates the user clicking the "back" button on their browser.
45	highlight("Locator")	Briefly changes the backgroundColor of the specified element yellow.
46	isAlertPresent()	Has an alert occurred?
47	isChecked("Locator")	Gets whether a toggle-button (checkbox/radio) is checked.
48	isConfirmationPresent()	Has confirm() been called?
49	isEditable("Locator")	Determines whether the specified input element is editable, ie hasn't been disabled.
50	isElementPresent("Locator")	Verifies that the specified element is somewhere on the page.
51	isPromptPresent()	Has a prompt occurred?
52	isSomethingSelected("Locator")	Determines whether some option in a drop-down menu is selected.
53	isTextPresent("Pattern")	Verifies that the specified text pattern appears somewhere on the rendered page shown to the user.
54	isVisible("Locator")	Determines if the specified element is visible.
55	open("URL")	Opens an URL in the test frame.
56	openWindow("URL","WindowID")	Opens a popup window (if a window with that ID isn't already open).
57	refresh()	Simulates the user clicking the "Refresh" button on their browser.
58	removeAllSelections("Locator")	Unselects all of the selected options in a multi-select element.
59	removeSelection("Locator","Option Locator")	Remove a selection from the set of selected options in a multi-select element using an option locator.

60	select("Select Locator","Option Locator")	Select an option from a drop-down using an option locator.
61	selectFrame("Locator")	Selects a frame within the current window.
62	selectWindow("WindowID")	Selects a popup window; once a popup window has been selected, all commands go to that window.
63	setSpeed("Value")	Set execution speed (i.e., set the millisecond length of a delay which will follow each selenium operation).
64	setTimeout("Time")	Specifies the amount of time that Selenium will wait for actions to complete.
65	start()	Launches the browser with a new Selenium session
66	stop()	Ends the test session, killing the browser
67	submit("Form Locator")	Submit the specified form.
68	type("Locator","Value")	Sets the value of an input field, as though you typed it in.
69	unCheck("Locator")	Uncheck a toggle-button (checkbox/radio)
70	waitForCondition("JavaScript","Timeout")	Runs the specified JavaScript snippet repeatedly until it evaluates to "true".
71	waitForFrameToLoad("Frame Address","Timeout")	Waits for a new frame to load.
72	waitForPageToLoad("Timeout")	Waits for a new page to load.
73	waitForPopUp("WindowID","Timeout")	Waits for a popup window to appear and load up.
74	windowFocus()	Gives focus to the currently selected window
75	windowMaximize()	Resize currently selected window to take up the entire screen

Handling Keyboard and Mouse:

S.No	Command	Description
1	altKeyDown()	Press the Alt key and hold it down until AltUp() is called or a new page is loaded.
2	altKeyUp()	Release the Alt key.
3	controlKeyDown()	Press the Control key and hold it

		down until ControlUp() is called or a new page is loaded.
4	controlKeyUp()	Release the Control key.
5	keyDown("Locator","Key Sequence")	Simulates a user pressing a key (without releasing it yet).
6	keyUp("Locator","Key Sequence")	Simulates a user releasing a key.
7	KeyPress("Locator","Key Sequence")	Simulates a user pressing and releasing a key.
8	metaKeyDown()	Press the meta key and hold it down until MetaUp() is called or a new page is loaded.
9	metaKeyUp()	Release the meta key.
10	mouseDown("Locator")	Simulates a user pressing the mouse button (without releasing it yet) on the specified element.
11	mouseDownAt("Locator","Coordinate String")	Simulates a user pressing the mouse button (without releasing it yet) at the specified location.
12	mouseMove("Locator")	Simulates a user pressing the mouse button (without releasing it yet) on the specified element.
13	mouseMoveAt("Locator","Coordinate String")	Simulates a user pressing the mouse button (without releasing it yet) on the specified element.
14	mouseOut("Locator")	Simulates a user moving the mouse pointer away from the specified element.
15	mouseOver("Locator")	Simulates a user hovering a mouse over the specified element.
16	mouseUp("Locator")	Simulates the event that occurs when the user releases the mouse button (i.e., stops holding the button down) on the specified element.
17	mouseUpAt("Locator","Coordinate String")	Simulates the event that occurs when the user releases the mouse button (i.e., stops holding the button down) at the specified location.
18	shiftKeyDown()	Press the shift key and hold it down until doShiftUp() is called or a new page is loaded.
19	shiftKeyUp()	Release the shift key.

S.No	Object/Property	Command
1	Text Box	getValue(), getText(), isEditable(), isVisible(), type()
2	List Box	getSelectedId(),getSelectedIndex(), , getSelectedLabel(),getSelectedValue(), getSelectOptions(), select()
3	Multi Select Element	getSelectedIds(),getSelectedIndexes(), getSelectedLabels(),getSelectedValues(),isSomethingSelected()
4	Radio Button	check(), click(), isChecked()
5	Check Box	Check(), click(),isChecked(), uncheck()
6	Button	click()
7	Link	click(), getAllLinks()
8	Text	getText(),getBodyText()
9	Tables and Cells	getTable()

Consider this feature matrix :

	Selenium IDE	Selenium RC	Selenium Core	Selenium Core HTA
Browser Support	Firefox Only	Many	All	IE Only
Requires Remote Installation	No	No	Yes	No
Supports HTTPS/SSL	Yes	Yes*	Yes	Yes
Supports Multiple Domains	Yes	Yes*	No	Yes
Requires Java	No	Yes	No	No
Saves Test Results to Disk	No**	Yes	No	Yes
Language Support	Selenese Only	Many	Selenese Only	Selenese Only

- *= experimental support is available in Selenium RC

- ** = theoretically possible, but not currently implemented