**Selenium** (3.1.0)- a suite of tools for automating web browsers. developed by Thoughtworks (Jason Huggins)

**Selenium suite** - Webdriver

- Selenium RC (selenium 1)

- IDE

- Grid

Later selenium 1 & Webdriver combined to Selenium 2 and Selenium 3

**Selenium IDE** - is an extension for chrome & Firefox browsers. By using this user can record the browser actions and can make test case out of it.

**Selenium RC** - controls the web browser through selenium server. It requires server to inject **js** into the browser

**two components**

- server

- client (our code , selenium commands)

**selenium commands -> Selenium server -> Web browser**

**RemoteWebDriver** is a class implements **Webdriver interface**

**Selenium Webdriver** - Webdriver drives the browser natively as the user would do. It runs with the help of OS.

**Webdriver is a interface** which extends the another interface called **SearchContext**

**Selenium Webdriver Architecture :**

Selenium client library --- JSON WIRE PROTOCOL---> Browser Drivers<----HTTP over HTTP server-----> Browser

- **Selenium client Library** - libraries for multi language java,python etc.

- **JSON WIRE PROTOCOL** - the library codes will be converted in to URL's in JASON format and then send to the server (every browser driver will have server)

- From **Browser drivers** request will be sent to the Browser over HTTP . Based on the request, actions will be done on the browser if it is a Post request then actions will be done on the browser . if it is a get request then response will be sent to the UI (IDE) from browser via Browser driver.

**Selenium Grid** - allows the user to run tests in different machines and different browser in parallel

**Different Drivers supported by Selenium**

ChromeDriver

FirefoxDriver

InternetExplorerDriver

SafariDriver

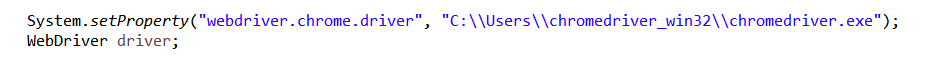
OperaDriver

HTMLUnitDriver (mock browser)

AndroidDriver (not available in official website)

IPhoneDriver (not available in official website)

**INITIALIZING WEBDRIVER**



**WebDriver driver;**  -------- webdriver is a interface which extends the another interface called SearchContext

**driver = new ChromeDriver();**  ------- chrome driver class which implements the interface Webdriver and this will open a empty browser.

**System.out.println(driver);** ----- this will print like this ChromeDriver: chrome on WINDOWS (2d22d94b4b56686aeb48750369028f55)

To open two chrome drivers:

With same reference

**driver = new ChromeDriver();**

**driver = new ChromeDriver();**

With different reference

**Driver1 = new ChromeDriver();**

**driver 2= new ChromeDriver();**

**CHROME OPTIONS** - used to manipulate various properties of Chrome driver

**ChromeOptions options = new ChromeOptions();**

options.addArguments("start-maximized");

options.addArguments("--incognito");

options.addArguments("--headless"); or options.setHeadless(true)

options.addArguments("disable-infobars");

options.addArgumentsrgument('--ignore-certificate-errors') -- ignores sss certification pop up error

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

**start-maximized:** Opens Chrome in maximize mode

**incognito:** Opens Chrome in incognito mode

**headless:** Opens Chrome in headless mode

**disable-extensions:** Disables existing extensions on Chrome browser

**disable-popup-blocking:** Disables pop-ups displayed on Chrome browser

**make-default-browser:** Makes Chrome default browser

**version:** Prints chrome browser version

**disable-infobars:** Prevents Chrome from displaying the notification 'Chrome is being controlled by automated software

**DESIRED CAPABILITIES -** is a class in selenium, used to modify the properties of web driver like browser name, browser version, path of browser driver in the system

**- It stores the capabilities as key-value pairs**

DesiredCapabilities capabilities = new DesiredCapabilities();

capabilities.setCapability("os\_version", "10");

capabilities.setCapability("browser", "Chrome");

capabilities.setCapability("browser\_version", "latest");

capabilities.setCapability("os", "Windows");

**driver = new RemoteWebDriver( new URL("https://www.browserstack.com/users/sign\_in"),capabilities); URL is grid url**

**COMMANDS IN WEBDRIVER**

-Browser command

-Navigation command

-Webelement command

**BROWSER COMMANDS**

driver.get(URL);

driver.getTitle(); ---Returns a null string if the page has no title (empty space will be printed)

driver.getCurrentUrl();

driver.getPageSource(); -- Returns the source code of the page as a String value

driver.close(); --- closes the current window

driver.quit(); --- closes all the tab

**NAVIGATE COMMANDS**

driver.navigate().to(URL);

driver.navigate().forward();

driver.navigate().back();

driver.navigate().refresh();

**WEBELEMENT COMMANDS**

**WebElement element = driver.findElement(By.id("UserName"));**

findElement - method from WebDriver interface

By - is a Abstract class (import org.openqa.selenium.By;)

element.clear();

element.sendKeys("text");

element.click();

element.isDisplayed();

element.isEnabled();

element.isSelected();

element.submit(); ----- driver.findElement(By.id("SubmitButton")).submit();

element.getText();

element.getTagName();

element.getCssValue("font-size");

element.getAttribute("id");

element.getSize();

element.getLocation();

**LOCATORS** - to locate an element in **DOM - document object model**

There are **8 types of locators** -- and all are **static methods of By Class**

**by ID**  driver.findElement(By.id("fname")).sendKeys("JavaTpoint");

**by Name**  driver.findElement(By.name("firstName")).sendKeys("Selenium Tutorial");

**by class name**  driver.findElement(By.className("Automation")).click();

**by Tag name** driver.findElement(By.tagName("input")).sendKeys("C++ Tutorial");

**by Link text**  driver.findElement(By.linkText("This is a Link")).click();

<a href="https://www.javatpoint.com/" > This is a Link </a> for multiple matches always first link will be clicked

**by Partial link text**  driver.findElement(By.partialLinkText("This is")).click();

Img links can be located using xpath or cssSelector

**by CSS** driver.findElement(By.cssSelector("Tag.Value of Class attribute")) driver.findElement(By.cssSelector("input.Automation"))

by Xpath

**Xpath** - used to locate dynamic elements or elemntes not able to locate by other locators.

Xpath=//tagname[@attribute='value']

**types:**

**Absolute XPath -** used to locate the element from the root node /html/body/div[2]

**Relative Xpath** - used to locate the element from the current node. //h4[1]//b[1]

**contains()** - checks for partial value Xpath=//\*[contains(@name,'btn')] btn is given for loginbtn

**OR expression** - //\*[@type='submit' or @name='btnReset']

**AND expression** - //input[@type='submit' and @name='btnLogin'] these 'or and' at attribute level

**starts-with()** - //label[starts-with(@id,'message')]

**text()** - exact match of the text //\*[text()='txt']

**To normalize space** - //\*[normalize-space(text())='txt']

**XPath axes** - to locate complex or dynamic elements

Following - **//following::**  //\*[@type='text']//following::input

Preceding - **//preceding::** //\*[@type='submit']//preceding::input

Ancestor -  **//ancestor::** //\*[text()='Enterprise Testing']//ancestor::div[1]

Child - **//child::**  //\*[@id='java\_technologies']//child::li

Following-sibling -  **//following-sibling::** //\*[@type='submit']//following-sibling::input

Parent - **//parent::** //\*[@id='rt-feature']//parent::div

Self - **//self::** //\*[@type='password']//self::input

Descendant -  **//descendant::** //\*[@id='rt-feature']//descendant::a[1] child and grand child

**special types:**

**SVG tag** - it is not possible to capture the svg (scalable vector graphics) using traditional xpath , so we need to use below syntax

eg: //\*[local-name()='svg'] or //\*[name()='svg']

**FIND ELEMENT** - used to uniquely identify a (one) web element

**Find Elements** - used to uniquely identify the list of web elements within the web page

**HANDLING CHECKBOX/RADIO BUTTONS**

to click - element.click();

to check whether selected or not - **element.isSelected();**

we can itreate through radio buttons

for(i=0;i<size;i++){

element.**get(i).click()**;} get(i) indicates second radio button (index number so starts from 0,1,2)

To check all check boxes

for(int i=0; i<=chkBox.size(); i++){

chkBox.get(i).click();

}

**HANDLING DROP-DOWNS**

WebElement element = driver.findElement(By.xpath("")); WebDriver - focuses on driving the browser WebElement - concentrates on interacting with a specific element that you've located

Select dropdown = new Select(element);

dropdown.selectByIndex(index); starts from 0

dropdown.selectByValue(arg0);

dropdown.selectByVisibleText(arg0);

**dropdown.isMultiple(); --- checks whether this dropdown element supports multiple selection at same time.**

List<WebElement> options = dropdown.getOptions(); --- returns all the options under select tag

WebElement firstSelectedOption = dropdown.getFirstSelectedOption();

List<WebElement> selectedOptions = select.getAllSelectedOptions();

to deselect

dropdown.deselectByValue(arg0);

dropdown.deselectByVisibleText(arg0);

**ACTIONs CLASS** - is a class in selenum to handel mouse and Keyboard actions , but **Action is Interface**

methods in selenium,

clickAndHold(Element) or clickAndHold(), for the later one we need to move to the element first using move to element

contextClick(Element) or contextClick() - Right Click Mouse Action

doubleClick(Element) or doubleClick()

dragAndDrop(source, target) or dragAndDrop(source, target)

keyDown(modifier\_key) - press modifier key - Keys.ALT, Keys.SHIFT, or Keys.CONTROL

keyUp(modifier \_key)

moveToElement(toElement) - Moves the mouse to the middle of the element

moveToElement(toElement, x offset, y offset) - used to handel sliders

release() - Releases the depressed left mouse button

sendKeys(onElement, charsequence)

**Actions act = new Actions(driver);**

act.click()

act.sendKeys(onElement, charsequence)

act.build() build() is used do generate composite action it is optional because it is built in perform()

act.perform() perform() is used perform the action

**To Handel drag and drop**

webelement source;

webelement target;

Actions act = new Actions(driver);

act.dragAndDrop(source, target).perform();

**To Handel slider**

webelement slider;

Actions act = new Actions(driver);

act.moveToElement(slider, 50, 0).perform();

slider.click();

**To select text in the textbox**

actions.keyDown(Keys.CONTROL); cursor need to be in the textbox

actions.sendKeys("a"); just change 'a' to 'c' for copy and 'v' for paste

actions.keyUp(Keys.CONTROL);

actions.build().perform();

**ALERT IN SELENIUM** - a small message box appears on the screen.

**types of alerts**

-**Simple Alert** - displays some information or warning on the screen with OK button

-**Prompt Alert** - asks some input from the user with text box , ok and cancel button

-**Confirmation Alert** - asks permission to do some type of operation with ok and cancel button

**methods to handel alerts**

void dismiss() - To click on the 'Cancel' button of the alert driver.switchTo().alert().dismiss();

void accept() - To click on the 'OK' button of the alert driver.switchTo().alert().accept();

String getText() - To capture the alert message driver.switchTo().alert().getText();

void sendKeys(String stringToSend) - To send some data to alert box driver.switchTo().alert().sendKeys("Text");

**WINDOW HANDLING** - switching to particular tab if multiple tabs are present

Driver.getWindowHandles(); - return the handels of all the windows

Driver.getWindowHandle(); - return the handel of the current window

If we print the handel it will be in format

**CDwindow-C69F9A6C9092AC4B8C77E2DDAC188E4C**

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

**we can only store handles in set , because getWindowHandles() will return Set<Strings> , if we try to store in list it will throw complie time error**

**y Set? to avoid enforcing order ... the order of the handels are random**

**To switch to newly created windowString**

parentW = driver.getWindowHandle();

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

for(String w : list) {

driver.switchTo().window(w);

}

driver.switchTo().window(parentW);

**To switch by title**

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

for(String w : list) {

if(driver.switchTo().window(w).getTitle().equals(title) {

break;

}

}

**To switch by index**

driver.switchTo().window( (String)driver.getWindowHandles().toArray()[index]);

**To create new tab**

js.executeScript("window.open()");

**ASSERTION** - determines the state of the application whether it is the same what we are expecting or not . Types Assert,Verify,waitFor

**Assert types**

**Hardassert** - stop the test execution when an assert statement fails - Hard asserts are the default type of asserts in TestNG

import org.testng.Assert;

important methods in hardassert

Assert.AssertFalse(condition);

Assert.AssertTrue(condition);

Assert.assertEquals(actual,expected);

AssertNotEquals(actual,expected,message);

Assert.assertNull(object);

Assert.assertNotNull(object);

Assert.fails();

we don't need to create object to acess the hardassert methods since all methods are static

try{

int i = 1/0;

}catch(Exception e) {

System.out.println("exe");

Assert.fail();

}

**Softassert** - test execution will not be stopped when an assert statement fails , and the subsequent statements will be executed . Soft assert does not include by default in TestNG

import org.testng.asserts.SoftAssert;

softassert have all the hardassert methods but they are non-static so we need to create object to access those methods

SoftAssert sa = new SoftAssert();

sa.assertAll(); -- this method only belogs to softassert , the execution will be continued until this statement for softassert

SoftAssert sa = new SoftAssert();

try{

int i = 1/0;

}catch(Exception e) {

System.out.println("exe");

sa.fail();

}

try{

int i = 1/0;

}catch(Exception e) {

sa.fail();

System.out.println("exe");

}

sa.assertAll();

**HANDLING iframe** -- its based on **method overloading** frames have different arguments

**By Index**

driver.switchTo().frame(0);

**By Name or Id**

driver.switchTo().frame("iframe1");

**By Web Element**

driver.switchTo().frame(WebElement);

**to switch to default frame**

driver.switchTo().defaultContent();

**To Handel multiple iframes**

WebElement wb ;

For(int i=0; i<wb.size();i++){

driver.switchTo().frame(i);

If(driver.findelements.size()>0){

Break;} }

**ROBOT CLASS** - used to automate Keyboard and Mouse events to control the OS native events like window pop ups

**Actions class - uses the WebDriver API and sends commands to a browser to perform actions**

important methods

Robot robot = new Robot();

robot.keyPress(KeyEvent.VK\_DOWN)

robot.mousePress(InputEvent.BUTTON3\_DOWN\_MASK)

robot.mouseMove(point.getX(), point.getY())

robot.keyRelease(KeyEvent.VK\_DOWN)

robot.mouseRelease(InputEvent.BUTTON3\_DOWN\_MASK)

**Synchronization in Selenium Webdriver**

we have two components during testing one is Application under test and test tool both will have different rate like application may load at different speed and tool execute script at some rate so both will be out of Synchronization, in order to achieve Synchronization, waits are used in selenium

**WAITS in selenium -** elements in web page will not load at same time , some element may load slowly to handle that waits in selenium used.

**two types of Synchronization**

**Unconditional** - will wait for a specified amount of time

Thread.Sleep();

**Conditional Synchronization** - will wait for the expected condition in a specified amount of time

**Types :**

**Implicit Wait** - used to tell the web driver to wait for a certain amount of time before it throws a "No Such Element Exception".

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

**Explicit Wait** - tells the driver to wait for a period of time until the expected condition achived, if not it will throw an exception based on the condition.

WebDriverWait wait = new WebDriverWait(driver, 20);

wait.until(ExpectedConditions.alertIsPresent());

**Methods in ExpectedConditions**

alertIsPresent()

elementToBeClickable(By)

elementToBeSelected(By)

invisibilityOfElementLocated(By) Stale element - element which is exit at first but after some changes in DOM , then that element might not available to interact for webdriver , so at that time Stale element exception will be displayed

presenceOfAllElementsLocatedBy(By)

presenceOfElementLocated(By)

textToBePresentInElement(WebElement, String)

textToBePresentInElementLocated(By, String)

titleIs(String)

visibilityOf(WebElement)

visibilityOfElementLocated(By)

visibilityOfAllElements(WebElement)

visibilityOfAllElementsLocatedBy(By)

**Fluent Wait** - will check for the element in a specified interval of time frequently until the maximum specified time is achived.

Wait wait = new FluentWait(WebDriver reference)

.withTimeout(timeout, SECONDS) - maximum time before throwing exception (30 sec)

.pollingEvery(timeout, SECONDS) - time frequency (5 sec) so checking frequency will be 6 times

.ignoring(Exception.class); - exception to avoid during every interval

**Exceptions in selenium** - **WebDriverException is the parent class which extends the runtime exceptions**

ElementNotVisibleException

NoSuchElementException - element not in the DOM

NoSuchFrameException

NoAlertPresentException

NoSuchWindowException

**StaleElementReferenceException -** when you find an element, the DOM gets modified then you reference the WebElement.

Eg:

WebElement we = driver.findElement(By.cssSelector("#valid"));

// you do something which alters the page or a javascript event alters the page we.click();

SessionNotFoundException - Webdriver is acting immediately after ‘quitting’ the browser

WebDriverException - Webdriver is acting immediately after ‘closing’ the browser

Timeout exception - occurs when a command takes longer than the wait time to avoid the ElementNotVisible Exception

**JAVASCRIPTEXECUTOR** - is an interface to execute Javascript through selenium

this interface only two methods

**executeScript()** - This method executes JavaScript in the context of the currently selected frame or window

**executeAsyncScript()**: This method executes an asynchronous piece of JavaScript in the context of the currently selected frame or window. In this Callback used as signal that the execution is completed.So Callback is the last argument.

**JavascriptExecutor js = (JavascriptExecutor) driver;**

**js.executeScript(Script,Arguments);**

script - This is the JavaScript that needs to execute.

Arguments – It is the arguments to the script. It's optional.

**SCROLLING**

**Scroll down by pixel**

JavascriptExecutor js = (JavascriptExecutor) driver; to scroll up give negative value in Y like -1000

js.executeScript("window.scrollBy(0,1000)");

**Scroll to visible element**

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].scrollIntoView();",Element );

**Scroll to bottom**

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

**Horizontal Scroll**

js.executeScript("arguments[0].scrollIntoView();",Element );

**diff between scroll and move to action is the move to element scrolls an element in the view and moves the mouse over the element**

**move to element - can be used when scrolling to particular element while java script have options like bottom, pixels, visible element.**

**Click by JS**

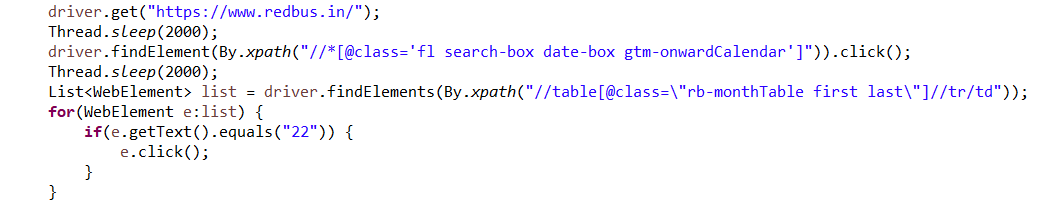
JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].click();", element);

**Entering text using js**

js.executeScript("arguments[0].value='ggg';", ele);

**Date Picker handling**



**FILE UPLOAD**

**Sendkeys method** - this can be used only If the element you send a file should be in the form <input type="file">

WebElement uploadElement = driver.findElement(By.id("uploadfile\_0"));

uploadElement.sendKeys("C:\\Users\\user\\Downloads\\P VINOTH KUMAR new resume.pdf");

**Auto It method**

Download and install Element Identifier and AutoIT editor

Open the site on which to do the operation

Element Identifier identifies the elements of file uploader window

Prepare AutoIT script in the editor with the help of Element identifier

use Autoit script in selenium webdriver script Runtime.getRuntime().exec("E:\\AutoIT\\FileUpload.exe");

**Robot class**

StringSelection strSelection = new StringSelection(path);

Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();

clipboard.setContents(strSelection, null);

Robot robot = new Robot();

robot.delay(300);

robot.keyPress(KeyEvent.VK\_ENTER);

robot.keyRelease(KeyEvent.VK\_ENTER);

robot.keyPress(KeyEvent.VK\_CONTROL);

robot.keyPress(KeyEvent.VK\_V);

robot.keyRelease(KeyEvent.VK\_V);

robot.keyRelease(KeyEvent.VK\_CONTROL);

robot.keyPress(KeyEvent.VK\_ENTER);

robot.delay(200);

robot.keyRelease(KeyEvent.VK\_ENTER);

**FILE DOWNLOAD**

**Using Robot Class**

**Using Chrome preferences**

ChromeOptions options = new ChromeOptions();

Map<String, Object> prefs = new HashMap<String, Object>();

prefs.put("download.prompt\_for\_download", false);

options.setExperimentalOption("prefs", prefs);

driver = new ChromeDriver(options);

**using java IO**

File des =new File("C:\\Users\\user\\Downloads\\text.pdf") ;

URL src = new URL("https://www.cdschools.org/cms/lib04/PA09000075/Centricity/Domain/81/BlenderBasics\_4thEdition2011.pdf");

FileUtils.copyURLToFile(src,des);

**SSL CERTIFICATE ERROR HANDLING IN CHROME**  - using desired capabilities

DesiredCapabilities handlSSLErr = DesiredCapabilities.chrome ()

handlSSLErr.setCapability (CapabilityType.ACCEPT\_SSL\_CERTS, true)

WebDriver driver = new ChromeDriver (handlSSLErr);

**SCREENSHOT**

TakesScreenshot ss = (TakesScreenshot) driver; TakesScreenshot is an interface

File srcimg = ss.getScreenshotAs(OutputType.FILE);

File des = new File ("C:\\Users\\user\\Downloads\\test.png");

FileUtils.copyFile(srcimg,des);

**Different way to refresh a wepage**

driver.navigate().refresh();

driver.findElement(By.locator).sendKeys(Keys.F5);

driver.get(driver.getCurrentUrl());

**COOKIES Handling**

A HTTP cookie is comprised of information about the user and their preferences. It stores information using a key-value pair

driver.manage().getCookies(); // Return The List of all Cookies

driver.manage().deleteAllCookies(); // Delete all cookies

driver.manage().addCookie(arg0); //Create and add the cookie

driver.manage().deleteCookie(arg0); // Delete specific cookie

**PROXY handling**

HTTP Proxy can also be used as a firewall between the Client and the server but can be used only for HTTP Requests. It act as a layer of protection which ask for Authentication.

Methods of handling Proxy

-AutoIT

-Using Alerts to handel HTTP Proxy authentication popup

driver.switchTo().alert().sendKeys("guru99");

driver.switchTo().alert().accept();

driver.switchTo().alert().sendKeys("guru99");

driver.switchTo().alert().accept();

**SELENIUM GRID**

Selenium Grid is a part of the Selenium Suite that specializes in 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

Using the RemoteWebDriver Object,

baseURL = "http://demo.guru99.com/test/guru99home/";

nodeURL = "http://192.168.43.223:4444/wd/hub";

DesiredCapabilities capability = DesiredCapabilities.chrome();

capability.setBrowserName("chrome");

capability.setPlatform(Platform.WIN10);

driver = new RemoteWebDriver(new URL(nodeURL), capability);

Questions,

What is the use of xpath ?

What are different types of locators ?

What is the difference between Assert and Verify?

What is the alternate way to click on login button?

How do you verify if the checkbox/radio is checked or not ?

How do you handle alert pop-up ?

How do you launch IE/chrome browser?

WebDriver driver = new InternetExplorerDriver();

How to perform right click using WebDriver?

Actions act = new Actions(driver); // where driver is WebDriver type

act.moveToElement(webElement).perform();

act.contextClick().perform();

How do perform drag and drop using WebDriver?

Give the example for method overload in WebDriver

frame(string), frame(int), frame(WebElement)

How do you upload a file?

How do you click on a menu item in a drop down menu?

How do you simulate browser back and forward

How do you get the current page URL ?

What is the difference between ‘/’ and ‘//’ ?

What is the difference between findElement and findElements?

How do you achieve synchronization in WebDriver ?

implicit wait

How to get typed text from a textbox ?

String typedText = driver.findElement(By.xpath("xpath of box")).getAttribute("value"));

What are the different exceptions you got when working with WebDriver ?

What are the languages supported by WebDriver ? Python, Ruby, C# and Java

How to invoke an application in webdriver ?

driver.get(“url”); or driver.navigate().to(“url”);

What is Selenium Grid ?

How to get the number of frames on a page ?

List <WebElement> framesList = driver.findElements(By.xpath("//iframe"));

int numOfFrames = frameList.size();

Which is the package which is to be imported while working with WebDriver ?

org.openqa.selenium

How to check if an element is visible on the web page ?

How to check if a button is enabled on the page ?

How to check if a text is highlighted on the page ?

String color = driver.findElement(By.xpath("//a[text()='Shop']")).getCssValue("color");

String backcolor = driver.findElement(By.xpath("//a[text()='Shop']")).getCssValue("background-color");

System.out.println(color);

System.out.println(backcolor);

How do u get the width of the textbox ?

driver.findElement(By.xpath(“xpath of textbox ”)).getSize().getWidth();

driver.findElement(By.xpath(“xpath of textbox ”)).getSize().getHeight();

How do u get the attribute of the web element ?

How to check whether a text is underlined or not ?

use actions to move mouse over the text and use .getCssValue("text-decoration");

How to change the URL on a webpage using selenium web driver ?

driver.get(“url1”);

driver.get(“url2”);

What is the use of getOptions() method ?

What is the use of deSelectAll() method ?

Is WebElement an interface or a class ?

Which is the super interface of webdriver ? -- SearchContext

What is the difference b/w close() and quit()?

Can we enter text without using sendKeys() ?

There is a scenario whenever “Assert.assertEquals()” function fails automatically it has to take screenshot. How can you achieve this ?

EventFiringWebDriver eDriver=new EventFiringWebDriver(driver);

File srcFile = eDriver.getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(srcFile, new File(imgPath));

How do you handle https website in selenium ?

options.addArguments('--ignore-certificate-errors')

How to login into any site if its showing any authetication popup for user name and pass ?

pass the username and password with url.

Syntax- http://username:password@url

How do you send ENTER/TAB keys in WebDriver ?

act.sendKeys(Keys.ENTER);

How to type text in a new line inside a text area ?

webelement.sendKeys(“Sanjay\_Line1.\n Sanjay\_Line2.”);

How to perform double click using WebDriver ? using actions act.doubleClick(webelement);

How to press Shift+Tab ?

Count the number of links in a page.

driver.findElements(By.tagName(“a”)).size();

Which one is better xpath or CSS -- xpath

How will you handle dynamic elements ? --- relative xpath

What do you mean by Selenese? Selenium IDE Commands are known as "Selenese"

types - Actions, Accessor, Assertions

How many types of WebDriver API's are available in Selenium?

AndroidDriver

ChromeDriver

EventFiringWebDriver

FirefoxDriver

HtmlUnitDriver

InternetExplorerDriver

iPhoneDriver

iPhoneSimulatorDriver

RemoteWebDriver

What is the difference between type keys and type commands? this available only in WebDriverBackedSelenium

TypeKeys() simulates the keystroke

Type just inputs the value

How to click on a hyper link using linkText?

driver.findElement(By.linkText("Google")).click();

test types that are supported by Selenium?

Functional and Regression

While using click command can you use screen coordinate? using clickAt(locator, coordinates) its selenium IDE command

What are the four parameter you have to pass in Selenium RC?

Host

Port number

Browser

URL

What is the difference between setSpeed() and sleep() methods?

Selenium.setSpeed("1000") - used to pause the driver for specific period of time for every selenium command

Thread.Sleep("1000") - used to pause the driver for specific period of time only for the current command.

Explain how to assert text of webpage using selenium 2.0 ?

String text = el.getText();

Assert.assertEquals(“Element Text”, text);

Which web driver implementation is fastest?

HTMLUnit Driver implementation is fastest, HTMLUnitDriver does not execute tests on browser but plain http request, which is far quick than launching a browser and executing tests