# Selenium with Java

<https://www.toolsqa.com/selenium-webdriver/selenium-tutorial/>

**Maven Project**

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. Using maven we can build and manage any Java based project.

WebDriver manager method is used to auto download latest version of Webdriver

WebDriverManager.*chromedriver*().setup();

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

Interface local variable = keyword constructor = class name

Create object for chromedriver class by calling chromedriver constructor new ChromeDriver will open the browser to perform certain actions for that we have instance for that browser store some variables and use those variables to perform certain actions it's called driver variable

We are creating a ChromeDriver instance and storing it in a variable called driver, which is a type of WebDriver interface

Now you can use it with this method

WebDriver driver = WebDriverManager.*chromedriver*().create();

**Desired Capabilities**

Add the Browser changes into the Selenium Webdriver when we run the Selenium script like a plugin, Zoom mode, popup excess and unauthorized excess web pages with the use of the Desired Capabilities class. It is common for all the browsers

DesiredCapabilities caps = new DesiredCapabilities();

caps.setAcceptInsecureCerts(true);

caps.setCapability();

* browserName
* browserVersion (replaces version)
* platformName (replaces platform)
* acceptInsecureCerts
* pageLoadStrategy
* proxy
* timeouts
* unhandledPromptBehavior

**ChromeOptions**

It has all the features of the desired capabilities class and some extra features for the Chrome browser such as Incognito window which is not available in Firefox and Edge browsers. So, every browser provides one particular class it’s called the ChromeOptions class. Add multiple arguments to customize the chrome windows

ChromeOptions options = new ChromeOptions();

options

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

options.addArguments("start-maximized");

Thread.*sleep*(2000);

options.addArguments("incognito");

ChromeOptions option = new ChromeOptions();

option.AddArgument("test-type");

option.AddArgument("start-maximized");

option.AddArgument("window-size=1920,1080"); option.AddArgument("enable-precise-memory-info"); option.AddArgument("disable-popup-blocking");

option.AddArgument("disable-default-apps");

option.AddArgument("test-type=browser");

## Selenium Locators

Locators are the way to identify an HTML element on a web page, and almost all UI automation tools provide the capability to use locators for the identification of HTML elements on a web page.

Following the same trend, Selenium also possesses the ability to use "Locators" for the identification of HTML elements and is popularly known as "Selenium Locators".

## What are Locators in Selenium?

Locators are one of the essential components of Selenium infrastructure, which help Selenium scripts in uniquely identifying the WebElements(such as text box, button, etc.) present on the web page.

## DOM (Document Object Model)

Represent the content of the HTML and XML documents as a tree structure

Can easily read, access, and update the content of the document

Is a programming Interface (API)

DOM is the object-oriented representation of the web page, which can be modified with a scripting language such as Javascript

## How to locate a web element in DOM?

1. DOM can access by clicking on the F12 button or right-click and select the Inspect option.
2. DOM structure web page opens the Element page
3. how do we find the web element in the DOM? Click on the "Mouse Icon" arrow and then select the web element on the web page. It will automatically highlight the corresponding HTML element in the DOM.

## What locators are supported by Selenium?

To access all the locators using “By” class and access the several methods className, cssSelector, id, linkText, partialLinkTest, tagName, XPath

By class locators

[By.ByClassName](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByClassName.html), [By.ByCssSelector](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByCssSelector.html), [By.ById](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ById.html), [By.ByLinkText](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByLinkText.html), [By.ByName](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByName.html), [By.ByPartialLinkText](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByPartialLinkText.html), [By.ByTagName](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByTagName.html), [By.ByXPath](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.ByXPath.html), [ByAll](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/support/pagefactory/ByAll.html), [ByChained](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/support/pagefactory/ByChained.html), [ByIdOrName](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/support/ByIdOrName.html), [RelativeLocator.RelativeBy](https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/support/locators/RelativeLocator.RelativeBy.html)

## How to use locators to find web elements with Selenium?

### How to locate a web element by using the "id" attribute?

"ID" as a locator is one of the most common ways of identifying elements on a web page. According to [W3C](https://www.w3.org/), an ID for a web element always needs to be unique.

By.id("firstName");

## How to locate a web element by using the "name" attribute?

the "By" class provides the "name" method, which accepts the value of the "name" attribute of the web element. So, using this, we can locate a web element that has a unique name attribute.

By.name("gender");

## How to locate a web element by using the "ClassName" attribute?

If we look at the *DOM* structure of the form, we can see the following snippet enclosing the entire form.

By.className("practice-form-wrapper");

## How to locate a web element by using the "LinkText" and "partial link text" attributes?

*LinkText* and *partialLinkText* are both quite similar in functionality and locate web elements by using hyperlink texts. We can only use them for elements containing *anchor(<a>) tags*.

By.linkText("Home");

By.partialLinkText("Ho");

## How to locate a web element by using the "TagName"?

This locator type uses *tag names* to identify elements in a web page. The *tag name* is the *HTML* tag, such as \*\*\*input, div, anchor tag, button, etc.

By.tagName("a");

## How to locate a web element by using the "CSS selector"?

*CSS or Cascading style sheets* are used extensively to style webpages and hence can be an effective medium to locate various web elements. These days most web pages are dynamically designed. Thus it's quite challenging to get a unique id, name, or class to locate an element.

css=(HTML Page)[Attribute=Value]

By.cssSelector("input[id= ‘userName’]");

## How to locate a web element by using the "XPath" attribute?

*XPath* uses the *XML* expression to locate an element on the webpage. Similar to CSS selectors, Xpath is quite useful in locating dynamic elements on a webpage.

//tag\_name[@attribute\_value]

By.xpath("//input[@id='userName']");

XPath = //tag\_name[@Attribute\_name = “Value of attribute”]

## Relative Locator

### above()

WebElement passwordField = driver.findElement(By.id(“password”));

WebElement emailaddressField = driver.findElement(By.withTagName(“input”).above(passwordField))

### below()

WebElement emailaddressField = driver.findElement(By.id(“emailaddress”));

WebElement passwordField = driver.findElement(By.withTagName(“input”).below(emailaddressField));

### toLeftOf()

By cancelLocator = RelativeLocator.with(By.tagName("button")).toLeftOf(By.id("submit"));

### toRightOf()

By submitLocator = RelativeLocator.with(By.tagName("button")).toRightOf(By.id("cancel"));

## What are the different types of XPaths in Selenium?

Selenium uses two strategies to locate elements using XPaths :

* Locating a web element using an Absolute XPath.
* Locating a web element using Relative XPath.

Let's understand both of these types in the following sections:

### What is Absolute XPath in Selenium?

Absolute XPath is the direct way of finding the element.

Moreover, it starts from the first/root node of the XML/HTML document and goes all the way to the required node following one node at a time.

/html/body/div/header/a/img

### What is Relative XPath in Selenium?

Relative XPath starts from any node inside the HTML DOM; it need not start from the root node. It beings with a double-forward slash.

It allowed us to trace the whole path and trace the element's exact position on the page. But imagine after some time if one more node changes, this previous absolute *XPath* will become redundant, so this is where *relative* *XPath* comes in handy.

//img[@src = "/images/Toolsqa.jpg"]

As we can see, relative XPath can start from any node. It also overcomes the fragile nature of Absolute XPath. Even if any preceding element changes or removes, it will not affect the Relative XPath.

**XPath Axes**

Relationship between nodes

Ancestor -> Parent and Grand Parent -> ancestor , ancestor or self

Parent

Sibling -> preceding-sibling -> following-sibling

Child

Descendant -> Child and Grand Child -> descendant , descendant or self

Big brother -> Preceding -sibling

Small brother -> following-sibling

Ancestor -> parent and grand parent

Ancestor or self -> Me, parent and grand parent

Descendant -> child and grand child

Descendant or self -> Me , child and grand child

## What is WebElement?

WebElement represents an HTML element. HTML documents are made up by HTML elements. HTML elements are written with a start tag, with an end tag, with the content in between: <tagname> content </tagname>

The HTML element is everything from the start tag to the end tag: <p> My first HTML paragraph. </p>every method of the WebDriver either returns something or returns void(which means return nothing). In the same way, the findElement command of WebDriver returns WebElement.

So, to get the WebElement object to write the below statement:

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

If you write the element dot then you can access the actions

WebElement can be of any type it can be a Text, Link, Radio Button, Drop Down, WebTable, or any HTML element.

### Clear Command

clear( ): void - If this element is a text entry element, this will clear the value. This method accepts nothing as a parameter and returns nothing.

**Command - element.clear();**

This method works fine with text entry elements like *INPUT* and *TEXTAREA* elements.

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

element.clear();

//Or can be written as

driver.findElement(By.id("UserName")).clear();

### SendKeys Command

sendKeys(CharSequence... keysToSend ): void - This simulates typing into an element, which may set its value. This method accepts CharSequence as a parameter and returns nothing.

**Command - element.sendKeys("text");**

This method works fine with text entry elements like INPUT and TEXTAREA elements.

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

element.sendKeys("ToolsQA");

//Or can be written as

driver.findElement(By.id("UserName")).sendKeys("ToolsQA");

### Click Command

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

### IsDisplayed Command

boolean staus = driver.findElement(By.id("UserName")).isDisplayed();

### IsEnabled Command

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

boolean status = element.isEnabled();

### IsSelected Command

WebElement element = driver.findElement(By.id("Sex-Male"));

boolean status = element.isSelected();

### Submit Command

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

element.submit();

### GetText Command

WebElement element = driver.findElement(By.xpath("anyLink"));

String linkText = element.getText();

### getTagName Command

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

String tagName = element.getTagName();

### getCssValue Command

Command - element.getCssValue();

### getAttribute Command

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

String attValue = element.getAttribute("id"); //This will return "SubmitButton"

### getSize Command

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

Dimension dimensions = element.getSize();

System.out.println(“Height :” + dimensions.height + ”Width : "+ dimensions.width);

### getLocation Command

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

Point point = element.getLocation();

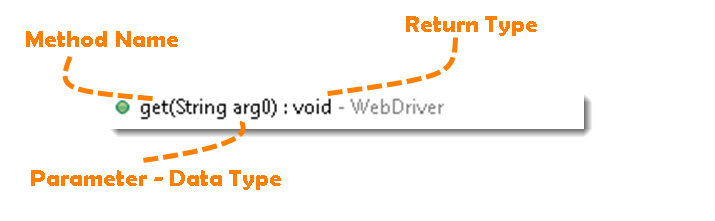
System.out.println("X cordinate : " + point.x + "Y cordinate: " + point.y);

### Chrome driver

### ChromeDriver is the communication medium that allows users to run their Selenium tests on the Chrome browsers.

## Browser Command

Methods followed by Object keyword are the generic methods gets from Object Class in Java. You will find these methods for every object of java language.



### Get Title Command - How to get the Title of the Webpage in Selenium?

Command - driver.getTitle();

### Get Current URL Command - How to read the URL of the Webpage in Selenium?

Command - driver.getCurrentUrl();

### Get Page Source Command - How to read the page source of the WebPage in Selenium?

Command - driver.getPageSource();

### Close Command - How to close the Browser in Selenium?

Command - driver.close();

### Quit Command - How to close all the Browser's Window in Selenium?

Command - driver.quit();

## Selenium Webdriver - Browser Navigation Command

### Navigate To Command - How to Navigate to URL or How to open a webpage in Selenium Browser?

Command - driver.navigate().to(appUrl);

driver.navigate().to("[https://www.DemoQA.com](https://www.demoqa.com)");

### Forward Command - How to browser Forward in Selenium Browser?

Takes you forward by one page on the browser's history.

Command - driver.navigate().forward();

### Back Command - How to browse backward in Selenium Browser?

Takes youback by one page on the browser's history.

driver.navigate().back();

### Refresh Command - How to Refresh Selenium Browser?

refresh(): void - This method Refresh the current page. It neither accepts nor returns anything.

Command - driver.navigate().refresh(); Perform the same function as pressing F5 in the browser.

Perform the same function as pressing F5 in the browser.

### Find elements using Selenium WebDriver?

To interact with WebElements, we first have to find or locate these elements on the webpage. We can find elements on a web page by specifying the attributes such as the Id of the element or the class name of the element and other parameters. These alternatives using which we can find elements on a webpage are called [locator strategies.](https://www.toolsqa.com/selenium-webdriver/selenium-locators/)

### Why do we need to find an element in Selenium?

### We know that we use Selenium mostly for UI testing of a web-based application. Since we need to perform automatic feature interaction with the web page, we need to locate web elements so that we can trigger some JavaScript events on web elements like click, select, enter, etc.

### How to find elements in Selenium?

findElement(): This method uniquely finds a web element on the web page.

findElements(): This method finds a list of web elements on the web page

## Action Class

There are complex interactions like Drag-n-Drop and Double-click which cannot be done by simple WebElement commands. To handle those types of advanced actions we have the Actions class in Selenium.

### What is the Actions class in Selenium?

The user-facing API for emulating complex user gestures. Use this class rather than using the Keyboard or Mouse directly.

Implements the builder pattern: Builds a CompositeAction containing all actions specified by the method calls

### Actions Class

Actions class is a collection of individual Action that you want to perform. For e.g. we may want to perform a mouse click on an element. In this case, we are looking at two different Actions

1. Moving the mouse pointer to the element
2. Clicking on the element

Collection of such Action is represented by the Actions class. There is a huge collection of methods available in the Actions class.

### What is Action Class?

single user interaction to perform the series of action items build by the Actions class. It contains one of the most widely used methods **perform().**

1.Import package:

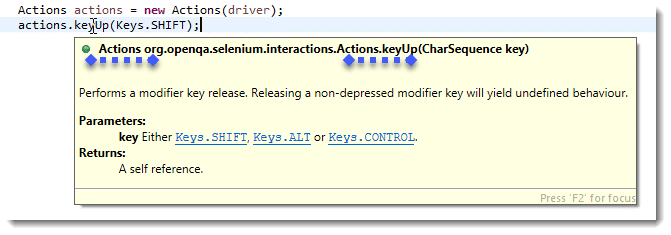
import org.openqa.selenium.interactions.Actions;

2. Instantiate Actions class:

Actions actions = new Actions(webdriver object);

*3*. Generate actions sequence:

* Pressing Shift Key: Actions Class Method => keyDown
* Sending the desired text: Actions Class Method => sendKeys
* Releasing Shift key: Actions Class Method => keyUp



actions.keyDown(element,Keys.SHIFT).sendKeys("TextToBeConvertAndSendInUpperCase").keyUp(Keys. SHIFT);

4. Build the actions sequence:

Action action = actions.build();

5. Perform actions sequence:

action.perform();

The keyDown method performs a modifier key press after focusing on an element, whereas keyUp method releases a modifier key pressed

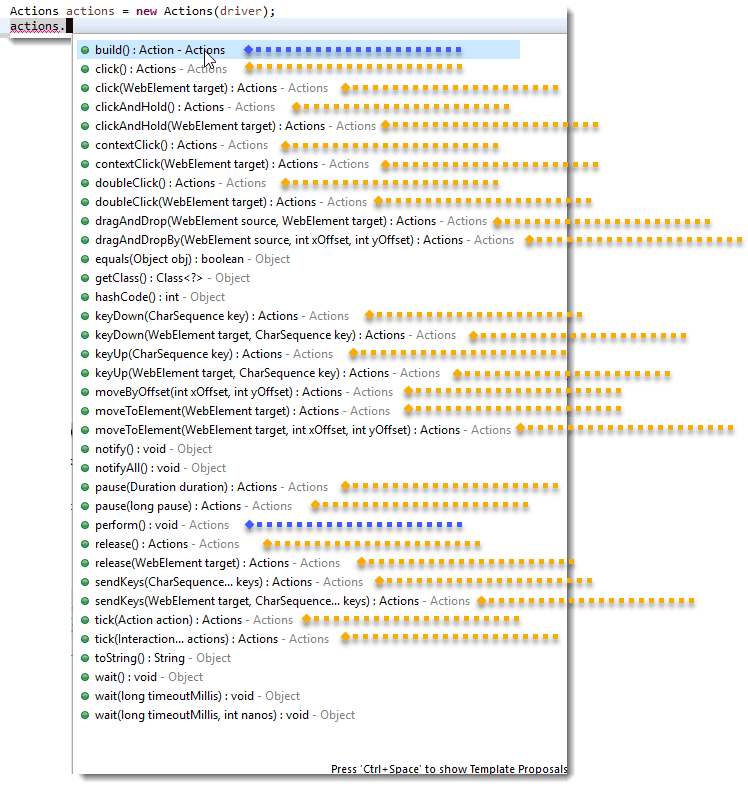
A modifier key is a key that modifies the action of another key when the two are pressed together like Shift, Control & Alt.

### Different Methods for performing :

* keyDown(modifier key): Performs a modifier key press.
* sendKeys(keys to send ): Sends keys to the active web element.
* keyUp(modifier key): Performs a modifier key release.

### Different Methods for performing Mouse Events:

* click(): Clicks at the current mouse location.
* doubleClick(): Performs a double-click at the current mouse location.
* contextClick(): Performs a context-click at the middle of the given element.
* clickAndHold(): Clicks (without releasing) in the middle of the given element.
* dragAndDrop(source, target): Click-and-hold at the location of the source element, moves to the location of the target element
* dragAndDropBy(source, xOffset, yOffset): Click-and-hold at the location of the source element, moves by a given offset
* moveByOffset(x-offset, y-offset): Moves the mouse from its current position (or 0,0) by the given offset
* moveToElement(toElement): Moves the mouse to the middle of the element
* release(): Releases the depressed left mouse button at the current mouse location



## How to Right Click in Selenium using Action Class?

To perform the right-click action through a Selenium script, WebDriver API does not have the capability for right-click commands like other Action commands: click, sendKeys.

That is where the Action class comes into play by providing various important methods to simulate user actions. And one of the most commonly used methods of the class is contextClick(WebElement), which is used to perform the Right-Click action.

Actions actions = new Actions(driver);

WebElement webElement = driver.findElement(Any By strategy & locator);

actions.contextClick(webElement).perform();

contectClick(): contextClick() method first performs mouseMove to the middle of the element location.

Build: build() method is used to generate a composite action containing all actions.

### Practice Exercise to Perform Right Click using Action Class in Selenium

//Instantiate Action Class

Actions actions = new Actions(driver);

//Retrieve WebElement to perform right click

WebElement btnElement = driver.findElement(By.id("rightClickBtn"));

//Right Click the button to display Context Menu&nbsp;

actions.contextClick(btnElement).perform();

System.out.println("Right click Context Menu displayed");

//Following code is to select item from context menu which gets open up on right click, this differs

//depending upon your application specific test case:

//Select and click 'Copy me' i.e. 2nd option in Context menu

WebElement elementOpen = driver.findElement(By.xpath(".//div[@id='rightclickItem']/div[1]"));

elementOpen.click();

// Accept the Alert

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

System.out.println("Right click Alert Accepted");

// Close the main window

driver.close();

## How to Double Click in Selenium using Action Class?

Double-click is a frequently used user action. The most common use of double click happens in File Explorer E.g. In File Explorer, any Folder or File in a folder can be opened by double-clicking on it.

Actions actions = new Actions(driver);

WebElement webElement = driver.findElement(Any By strategy);

actions.doubleClick(webElement).perform();

doubleClick() method also follows the same process of Move to Element >> Build >> Perform which is same as for right-click.

### Practice Exercise to Perform Double Click using Action Class in Selenium

//Instantiate Action Class

Actions actions = new Actions(driver);

//Retrieve WebElement to perform double click

WebElement btnElement = driver.findElement(By.id("doubleClickBtn"));

//Double Click the button

actions.doubleClick(btnElement).perform();

System.out.println("Button is double clicked");

//Following code just click on OK button on alert , this differs

//depending upon application(under test) specific test case

// Accept the Alert

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

System.out.println("Double click Alert Accepted");

//Close the main window

driver.close();

## Drag and Drop in Selenium

This is an action performed with a mouse when a user moves (drags) a web element and then places (drops) it into an alternate area.To perform the drag-drop action through a Selenium script, there is no direct drag-drop method available in WebElement interface.

Actions methods for drag & drop:

dragAndDrop(WebElementsource, WebElement target)

dragAndDropBy(WebElementsource, int xOffset, int yOffset)

### 1.dragAndDrop(WebElementsource, WebElement target)

Actions actions = new Actions(driver);

WebElement source = driver.findElement(Any By strategy & locator);

WebElement target = driver.findElement(Any By strategy & locator);

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

WebElement sourse = a.findElement(By.xpath("(//div[@class='c-day-content'])[26]"));

source.click();

WebElement targets = a.findElement(By.xpath("(//div[@class='c-day-content'])[31]"));

targets.click();

action.dragAndDrop(sourse, targets).build().perform();

Thread.sleep(2000);

* Click And Hold Action: dragAndDrop() method first performs click-and-hold at the location of the source element
* Move Mouse Action: Then source element gets moved to the location of the target element
* Button Release Action: Finally, it releases the mouse
* Build: build() method is used to generate a composite action containing all actions. But if you observe, we have not invoked it in our above command. The build is executed in the perform method internally
* Perform: perform() method performs the actions we have specified. But before that, it internally invokes build() method first. After the build, the action is performed

### Practice Exercise to Perform Drag and Drop using Actions Class in Selenium

//Actions class method to drag and drop

Actions builder = new Actions(driver);

WebElement from = driver.findElement(By.id("draggable"));

WebElement to = driver.findElement(By.id("droppable"));

//Perform drag and drop

builder.dragAndDrop(from, to).perform();

//verify text changed in to 'Drop here' box

String textTo = to.getText();

if(textTo.equals("Dropped!")) {

System.out.println("PASS: Source is dropped to target as expected");

}else {

System.out.println("FAIL: Source couldn't be dropped to target as expected");

}

driver.close();

### 2. dragAndDropBy(WebElement source, int xOffset, int yOffset):

This method clicks & holds the source element and moves by a given offset, then releases the mouse. Offsets are defined by x & y.

* xOffset is horizontal movement
* yOffset is a vertical movement

Actions actions = new Actions(driver);

WebElement source = driver.findElement(Any By strategy & locator);

the remaining two parameter values i.e. xOffset and yOffset are values in pixel.

For Example, if a xOffset value is set as 50, it means an object needs to be dragged and dropped by 50 pixels offset horizontal direction. Similarly, if a yOffset value is set as 50, it means an object needs to be dragged and dropped by 50 pixels offset vertical direction.

actions.dragAndDropBy(source, xOffset, yOffset).perform();

//Actions class method to drag and drop

Actions builder = new Actions(driver);

WebElement from = driver.findElement(By.id("draggable"));

WebElement to = driver.findElement(By.id("droppable"));

//Here, getting x and y offset to drop source object on target object location

//First, get x and y offset for from object

int xOffset1 = from.getLocation().getX();

int yOffset1 = from.getLocation().getY();

System.out.println("xOffset1--->"+xOffset1+" yOffset1--->"+yOffset1);

//Secondly, get x and y offset for to object

int xOffset = to.getLocation().getX();

int yOffset = to.getLocation().getY();

System.out.println("xOffset--->"+xOffset+" yOffset--->"+yOffset);

//Find the xOffset and yOffset difference to find x and y offset needed in which from object required to dragged and dropped

xOffset =(xOffset-xOffset1)+10;

yOffset=(yOffset-yOffset1)+20;

//Perform dragAndDropBy

builder.dragAndDropBy(from, xOffset,yOffset).perform();

//verify text changed in to 'Drop here' box

//Get text value of 'to' element

String textTo = to.getText();

if(textTo.equals("Dropped!")) {

System.out.println("PASS: Source is dropped at location as expected");

}else {

System.out.println("FAIL: Source couldn't be dropped at location as expected");

}

driver.close();

Final Code:

//Actions class method to drag and drop

Actions builder = new Actions(driver);

WebElement from = driver.findElement(By.id("draggable"));

//Perform dragAndDropBy

builder.dragAndDropBy(from, 100,100).perform();

System.out.println("Dropped");

## Mouse Hover action in Selenium

Mouse hover action is basically an action where a user places a mouse over a designated area like a hyperlink.

Actions class has provided for Mouse Hover action:

1. moveToElement(WebElement target)
2. moveToElement(WebElement target, int xOffset, int yOffset)

### 1. How to move the mouse cursor to the middle of the web element?

Actions actions = new Actions(driver);

WebElement target = driver.findElement(Any By strategy & locator);

actions.moveToElement(target).perform();

//Retrieve WebElement 'Music' to perform mouse hover

WebElement menuOption = driver.findElement(By.xpath(".//div[contains(text(),'Music')]"));

//Mouse hover menuOption 'Music'

actions.moveToElement(menuOption).perform();

System.out.println("Done Mouse hover on 'Music' from Menu");

//Now Select 'Rock' from sub menu which has got displayed on mouse hover of 'Music'

WebElement subMenuOption = driver.findElement(By.xpath(".//div[contains(text(),'Rock')]"));

//Mouse hover menuOption 'Rock'

actions.moveToElement(subMenuOption).perform();

System.out.println("Done Mouse hover on 'Rock' from Menu");

//Now , finally, it displays the desired menu list from which required option needs to be selected

//Now Select 'Alternative' from sub menu which has got displayed on mouse hover of 'Rock'

WebElement selectMenuOption = driver.findElement(By.xpath(".//div[contains(text(),'Alternative')]"));

selectMenuOption.click();

System.out.println("Selected 'Alternative' from Menu");

// Close the main window

driver.close();

### 2. How to move the mouse cursor to an element offset?

moveToElement(WebElement target, int xOffset, int yOffset): Moves the mouse to an offset from the top-left corner of the element.

Actions actions = new Actions(driver);

WebElement target = driver.findElement(Any By strategy & locator);

actions.moveToElement(target, xOffset, yOffset).perform();

#### Practice Exercise to Perform Mouse Hover using Actions Class in Selenium

//Instantiate Action Class

Actions actions = new Actions(driver);

//Retrieve WebElemnt 'slider' to perform mouse hover

WebElement slider = driver.findElement(By.id("slider"));

//Move mouse to x offset 50 i.e. in horizontal direction

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

slider.click();

System.out.println("Moved slider in horizontal directions");

## What are XPath Functions in Selenium?

A simple XPath may return more than one element. To overcome such scenarios, XPath in Selenium offers XPath functions that can write effective XPaths to identify elements uniquely.

### Xpath Contains() function

Contains() function is useful for finding dynamic web elements

//tag\_name[contains(@attribute,'value\_of\_attribute')]

Two parameters:-

* The attribute of the tag needs to validate for locating the web element.
* The Partial value of the attribute, which the attribute should contain.

//input[contains(@id, "userN")]

### XPath Starts-with() function

Imagine an element that has an attribute value that keeps on changing with every page load or page operation. Usually, these dynamic elements have few common starting characters, followed by random dynamic texts.

Starts-with() function is useful for finding dynamic web elements

Use to match the starting value of web element which remain static

//tag\_name[starts-with(@attribute,'Part\_of\_Attribute\_value')]

Two parameters:

* The attribute of the tag needs to validate for locating the web element.
* The Partial value of the attribute, which we expect from the attribute to start.

//input[starts-with(@placeholder,"Fu")]

### XPath Text() function

//tag\_name[text()='Text of the element'

* text() method is useful for finding the element with the exact text match.
* Here we directly used the text present on the web page to identify the element instead of using the attribute and values to recognize the same.

//label[text()=”Email”]

### AND & OR operators

The "and " operator combines two different conditions or attributes to identify any element from a webpage using XPath efficiently.

//tag\_name[@name = 'Name value' and @id = ‘ID value’]

If the web element has an ID attribute value then you can use id attribute details in the CSS selector by symbolized value #.

tagname#elementID

If the web element has a Class attribute value then you can use Class attribute details in the CSS selector by symbolized value DOT sign(.)

tagname.elementID

**Types Of CSS Selector**

* Simple CSS Selector
* Element
* ID
* Class
* Universal

<https://www.hyrtutorials.com/p/css-selectors-practice.html>

* Attribute CSS Selector

Attribute Name - input[placeholder]

Attribute Value - input[placeholder='First Name']

Attribute value - partial text - whole word - input[placeholder~='security']

Attribute value - partial text - text - input[placeholder\*='curi']

Attribute value - startswith - whole word - p[class|='top']

Attribute value - startswith - text - p[class^='top']

Attribute value - ends with - input[placeholder$='answer']

* Combinator CSS Selector

Descendant selector -> .container input

Child selector -> .container>select>option

Adjancent sibling selector -> input[placeholder='Enter your security question']+input

General sibling selector -> input[placeholder='Enter your security question']~a

* Pseudo CSS Selector

First-child .container>:first-child

Last-child .container>:last-child

Nth-child .container>:nth-child(4)

First-of-type .container>input:first-of-type

Last-of-type .container>input:last-of-type

nth-of-type .container>input:nth-of-type(8)

## Advance CSS selector

Syntax : tagName.classValue[AttributeName=’AttributeValue’]

tagName#idValue[AttributeName=’AttributeValue’]

## CSS selector SubString

Sub string matches are very helpful for identifying dynamic web elements with partial string matches.

3 important substring characters in the substrings

“^” Prefix of the text

“$” Suffix of the text

“\*” Substring of the text

## Finding Child and SubChild Elements

### Direct Child

Child combinator(>) is used to select direct child

tagName[AttributeName=’AttributeValue’] > tagName[AttributeName=’AttributeValue’]

select#country>option[value=’AU’]

### Child or Sub child

A descendant combinator() is used to select a Child or Sub child

tagName[AttributeName=’AtrributeValue’]

tagName[AttributeName=’AtrributeValue’]

### CSS Selector Next Sibling

Substrings are location by using (+) operator

Adjacent sibling combinator (+) separates two selectors and matches the first web elements with the second web elements only if it followed immediately the first web element and both are the child of the same parent node.

tagName[AttributeName=’AtrributeValue’] +

tagName[AttributeName=’AtrributeValue’]

## Basic method in Webdriver Interface

1. get() - load a new page in the current browser window
2. manage() - get the option interface
3. getCurrentURL() - get a string representing the current URL that the browser is looking at.
4. getTitle() - the title of the current node
5. navigate() - driver navigates to a browser history and navigates to a current URL
6. quit() - quit the driver with the associated window
7. close() - close the current windows, quitting the plan if it is the last web page of the window

## Dropdown in Selenium

### What is Select Class in Selenium?

Select select = new Select(WebElement webelement);

### How to select a value from a dropdown in Selenium?

* selectByIndex
* selectByValue
* selectByVisibleText

#### selectByIndex:

it accepts the index of the *dropdown* value, which needs to be selected. The index starts at 0.

// Create an object of the Select class

Select se = new Select(driver.findElement(By.xpath("//\*[@id='oldSelectMenu']")));

// Select the option by index

se.selectByIndex(3);

#### selectByValue

// Create an object of the Select class

Select se = new Select(driver.findElement(By.xpath("//\*[@id='oldSelectMenu']")));

// Select the option with the value "6"

se.selectByValue("6");

#### selectByVisibleText

// Create the object of the Select class

Select se = new Select(driver.findElement(By.xpath("//\*[@id='oldSelectMenu']")));

// Select the option using the visible text

se.selectByVisibleText("White");

### How to check whether the dropdown is Multi-Select?

Select oSel = new Select(driver.findElement(By.xpath(//\*[@id='cars']);

if(oSel.isMultiple()){

//Selecting multiple values by index

oSel.selectByIndex(1);

oSel.selectByIndex(2);

//Or selecting by values

oSel.selectByValue("Volvo");

oSel.selectByValue("Audi");

//Or selecting by visible text

oSel.selectByVisibleText("Volvo");

oSel.selectByVisibleText("Opel");

### How to get options from a dropdown in Selenium?

* getOptions()
* getFirstSelectedOption()
* getSelectedOptions()

#### getOptions

you need to get all the options in a dropdown or multi-select box. This is where you can use the getOptions() method of the Select class.

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get all the options of the dropdown

List<WebElement> options = select.getOptions();

#### getFirstSelectedOption()

This method returns the first selected option of the dropdown. If it is a single-select dropdown, this method will return the selected value of the dropdown, and if it is a multi-select dropdown, this method will return the first selected value of the dropdown.

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get the first selected option of the dropdown

WebElement firstSelectedOption = select.getFirstSelectedOption();

#### getAllSelectedOptions()

This method returns all the selected options of the dropdown. If it is a single-select dropdown, this method will return the only selected value of the dropdown, and if it is a multi-select dropdown, this method will return all the selected values of the dropdown.

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get all the selected options of the dropdown

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

### How to deselect a value from a dropdown in Selenium?

deselect method works only for Multi-Select. You can deselect pre-selected options from a Multi-select element using the different deselect methods discussed here.

* deselectAll()
* deselectByIndex()
* deselectByValue()
* deselectByVisibleText()

#### deselectAll

If there are a few options already selected in a dropdown, you can deselect all the options using the method deselectAll().

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect all the options

select.deselectAll();

#### deselectByIndex

selectByIndex() method, the Select class also provides the method to deselect an option from the dropdown using the deselectByIndex() method.

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect first value by index

select.deselectByIndex(1);

#### deselectByValue

selectByValue() method, the Select class also provides the method to deselect an option from the dropdown using the deselectByValue() method.

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect option with value "6"

select.deselectByValue("6");

#### deselectByVisibleText

selectByVisibleText() method, the Select class also provides the method to deselect an option from the dropdown using the deselectByVisibleText() method.

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect option with text "White"

select.deselectByVisibleText("White");

## Checkbox in Selenium

### What is a CheckBox?

a checkbox in HTML using <input type="checkbox"> tag. Any [locator strategy](https://www.toolsqa.com/selenium-webdriver/selenium-locators/) that uses [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) for locating web elements should use this tag and properties for recognizing the checkbox.

### How to handle a CheckBox in Selenium WebDriver?

#### How to locate and select a checkbox in Selenium using the ID locator?

If a checkbox has an id attribute that contains a unique value, then we can use the ID locator provided by the Selenium WebDriver for locating and selecting the element. To select a checkbox, the click operation needs to perform.

driver.findElement(By.id("hobbies-checkbox-1")).click();

#### How to locate and select a checkbox in Selenium using the XPath locator?

//Selecting the second checkbox using Xpath

driver.findElement(By.xpath("//label[text()='Reading']")).click();

#### How to locate and select a checkbox in Selenium using the CSS Selector locator?

//Selecting the first checkbox

driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']").click();

#### How to use the isSelected() method to validate if the CheckBox is selected?

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']"));

boolean isSelected = checkBoxElement.isSelected();

//performing click operation if element is not checked

if(isSelected == false) {

checkBoxElement.click();

}

#### How to use isDisplayed() method to validate if the CheckBox is displayed?

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']"));

boolean isDisplayed = checkBoxElement.isDisplayed();

// performing click operation if an element is displayed

if (isDisplayed == true) {

checkBoxElement.click();

}

#### How to use the isEnabled() method to validate if the CheckBox is enabled?

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']"));

boolean isEnabled = chckBxEnable.isEnabled();

// performing click operation if an element is enabled

if (isEnabled == true) {

checkBoxElement.click();

}

## Radio Button in Selenium

### What is a Radio Button?

A radio button in HTML is defined using <input> tag and an attribute "type", which will have the value "radio". So any [locator strategy](https://www.toolsqa.com/selenium-webdriver/selenium-locators/) that uses [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) for identifying and locating the elements will use the <input> tag for recognizing the radio buttons.

### How to select a Radio Button using Selenium WebDriver?

#### How to locate a radio button using an ID locator?

Locating and Clicking Radio Button By using IDdriver.findElement(By.id("yesRadio")).click();

#### How to locate a radio button using the name locator?

driver.findElement(By.name("like")).click();

#### How to locate a radio button using the XPath locator?

driver.findElement(By.xpath("//div/input[@id='yesRadio']")).click();

#### How to locate a radio button using the CSS Selector locator?

driver.findElement(By.cssSelector("input[id='yesRadio']")).click();

#### How to verify that the Radio Button is selected using Selenium isSelected() method?

WebElement radioElement = driver.findElement(By.id("impressiveRadio"));

boolean selectState = radioElement.isSelected();

//performing click operation only if element is not selected

if(selectState == false) {

radioElement.click();

}

#### How to verify that if the Radio Button is displayed using Selenium isDisplayed() method?

Validate Radio button using isDisplayed() methiod

WebElement radioElement = driver.findElement(By.id("impressiveRadio"));

boolean selectState = radioElement.isDisplayed();

//performing click operation only if element is not selected

if(selectState == false) {

radioElement.click();

}

#### How to verify that if the Radio Button is enabled using Selenium isEnabled() method?

WebElement radioElement = driver.findElement(By.id("noRadio"));

boolean selectState = radioElement.isEnabled();

//performing click operation only if element is not selected

if(selectState == false) {

radioElement.click();

}

## ToolTip in Selenium

* Click on the Inspect menu and then select the sources sub-menu
* Click on the F8 key pause the debugger
* After that click on the tooltip text and get the locator for that element
* Case One: When the tooltip is available in the 'title' attribute. Here, we can retrieve tooltip from By strategy

WebElement searchbox = driver.findElement(By.locator strategy);

String tooltipText = searchbox.getAttribute(“title”);

Syso(“Tooltip Text is : ”+tooltipText);

* Case Two: When the tooltip is available in 'div'. Here, we can retrieve tooltip using Actions class methods

Actions action = new Actions(driver);

WebElement mousehover = driver.findElement(By.locator strategy );

action.moveToElement(mousehover).build().perform();

WebElement tooltipmessage = driver.findElement(By.locators strategy);

Syso(“Tooltip message is : ”+tooltipmessage.getText());

## WAIT in Selenium

Wait command in selenium direct test scripts to pause for certain time before throwing “ElementNotVisibleExeception”.

### Types of Selenium Waits

Selenium wait commands has two sections:

* Implicit wait
* Explicit wait
* Fluent wait

#### Implicit wait

* Implicit wait tells the Webdriver maximum wait time when searching for element before throwing exception.
* Implicit wait is global it is applicable to all the webelement in the script
* Implicit wait applies to all the webelements you do not specify “Expected conditions ” on the element to be located.

Syntax:

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

We can apply implicit wait through three functions:

* implicitlyWait()
* pageLoadTimeout()
* setScriptTimeout()

#### pageLoadTimeout command in Selenium timeout

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);

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

#### setScriptTimeout command in Selenium timeout

driver.manage().timeouts().setScriptTimeout(15, TimeUnit.SECONDS);

((JavascriptExecutor) driver).executeScript("alert('hello world');");

((JavascriptExecutor) driver).executeAsyncScript("window.setTimeout(arguments[arguments.length - 1], 500);");

#### Explicit wait

* Explicit wait tells the Webdriver to halt the execution until a particular condition is met or the maximum time has passed
* Applicable for those webelements which are specified by the user
* Explicit wait require the expected conditions to be specified for the elements like “Wait Until Element Is Enabled”

two types of explicit wait commands:

* WebDriverWait
* FluentWait

#### WebDriverWait command in Selenium

WebElement firstResult = new WebDriverWait(driver, Duration.ofSeconds(10))

.until(ExpectedConditions.elementToBeClickable(By.xpath("//a/h3")));

#### Fluent wait in Selenium

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5)) .ignoring(NoSuchElementException.class);

### What is a window handle in Selenium?

A window handle stores the unique address of the browser windows.It is just a pointer to a window, whose return type is alphanumeric. The window handle in Selenium helps in handling multiple windows and child windows. Each browser will have a unique window handle value with which we can uniquely identify it.

#### What are the different methods used for window handling in Selenium?

#### getWindowHandle( )

getWindowHandles( )

switchto()

action

//Get handles of the windows

String mainWindowHandle = driver.getWindowHandle();

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

Iterator<String> iterator = allWindowHandles.iterator();

## What are Alerts/popups in Selenium?

Alerts are small popup boxes/windows which display the messages/notifications and notify the user with some information seeking some permission on certain kinds of operations. Additionally, we can also use them for warning purposes. Sometimes, the user can enter a few details in the alert box as well.

### What are the different types of Alerts/popups?

While automating any web application, [Selenium WebDriver](https://www.toolsqa.com/selenium-webdriver/selenium-tutorial/) may encounter alerts that can either be application dependant or the Operating system dependant on which the user is working.

#### Windows/OS Alerts:

Window-based alerts are system-generated alerts/popups. A sample operating system based alert will look as follows and are majorly called Dialog-Boxes

#### Web/Javascript /Browser-based Alerts:

Web/Browser based alerts are primarily called Javascript alerts and are those alerts that are browser dependant. These alerts are majorly called Popups.

### What are the various kinds of alerts provided by Web Applications?

#### Simple alert:

These alerts are just informational alerts and have an OK button on them. Users can click on the OK button after reading the message displayed on the alert box.

#### Prompt Alert:

In Prompt alerts, some input requirement is there from the user in the form of text needs to enter in the alert box. A prompt alert box is displayed like below, where the user can enter his/her username and press the OK button or Cancel the alert box without entering any details .

#### Confirmation Alert:

These alerts get some confirmation from the user in the form of accepting or dismissing the message box.

### How to handle Alerts/popups using Selenium WebDriver?

To switch the control from the parent window to the Alert window, the Selenium WebDriver provides the following command:

driver.switchTo( ).alert( );

For example, accepting the alert, dismissing the alert, getting the text from the alert window, writing some text on the alert window, and so on.

To handle Javascript alerts, Selenium WebDriver provides the package org.openqa.selenium.Alert and exposes the following methods:

* Void accept(): This method clicks on the 'OK' button of the alert box.

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

* Void dismiss(): We use this method when the 'Cancel' button clicks in the alert box.

driver.switchTo( ).alert( ).dismiss();

* String getText(): This method captures the message from the alert box.

driver.switchTo().alert().getText();

* Void sendKeys(String stringToSend): This method sends data to the alert box.

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

### How to handle a Simple Alert using Selenium WebDriver?

// This step will result in an alert on screen

driver.findElement(By.id("alertButton")).click();

Alert simpleAlert = driver.switchTo().alert();

simpleAlert.accept();

### How to handle a Prompt Alert using Selenium WebDriver?

// This step will result in an alert on screen

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

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

Alert promptAlert = driver.switchTo().alert();

String alertText = promptAlert.getText();

System.out.println("Alert text is " + alertText);

//Send some text to the alert

promptAlert.sendKeys("Test User");

promptAlert.accept();

### How to handle a Confirmation Alert using Selenium WebDriver?

// This step will result in an alert on screen

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

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

Alert confirmationAlert = driver.switchTo().alert();

String alertText = confirmationAlert.getText();

System.out.println("Alert text is " + alertText);

confirmationAlert.accept();

### How to handle unexpected Alerts using Selenium WebDriver?

try {

driver.findElement(By.id("timerAlertButton")).click();

WebDriverWait wait = new WebDriverWait(driver,10);

wait.until(ExpectedConditions.alertIsPresent());

Alert simpleAlert = driver.switchTo().alert();

simpleAlert.accept();

System.out.println("Unexpected alert accepted");

} catch (Exception e) {

System.out.println("unexpected alert not present");

}

driver.quit();

## How to Handle IFrame / IFrames with Selenium WebDriver

* **switchTo.frame(int frameNumber):** Pass the frame index and driver will switch to that frame.
* **switchTo.frame(string frameNameOrId):** Pass the frame element Name or ID and driver will switch to that frame.
* **switchTo.frame(WebElement frameElement):** Pass the frame web element and driver will switch to that frame.

//By executing a java script

JavascriptExecutor exe = (JavascriptExecutor) driver;

Integer numberOfFrames = Integer.parseInt(exe.executeScript("return window.length").toString());

System.out.println("Number of iframes on the page are " + numberOfFrames);

//By finding all the web elements using iframe tag

List<WebElement> iframeElements = driver.findElements(By.tagName("iframe"));

System.out.println("The total number of iframes are " + iframeElements.size());

### Switch to Frames by Index

//Switch by Index

driver.switchTo().frame(0);

driver.quit();

### Switch to Frames by Name

//Switch by frame name

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

driver.quit();

### Switch to Frame by ID

//Switch by frame ID

driver.switchTo().frame("IF1");

driver.quit();

### Switch to Frame by WebElement

### //First find the element using any of locator stratedgy

### WebElement iframeElement = driver.findElement(By.id("IF1"));

### 

### //now use the switch command

### driver.switchTo().frame(iframeElement);

### driver.quit();

### Switching back to the Main page from Frame

//First find the element using any of locator stratedgy

WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command

driver.switchTo().frame(0);

//Do all the required tasks in the frame 0

//Switch back to the main window

driver.switchTo().defaultContent();

driver.quit();

### What are iframes?

An iframe is a space on the web page that embeds different kinds of media like images, documents, and videos inside the main web page.

<iframe src="URL"></iframe>

### What is the difference between a frame and an iframe?

Frames are the HTML tags that divide the browser's window into multiple parts, where each part can load a separate HTML document. Each frame is assigned its webpage. The <frame> tag denotes a frame, and all the frames are present in a <frameset> tag in HTML. A frameset is defined as a collection of frames in a browser window

### How to automate iframes using Selenium WebDriver?

Selenium first needs to switch the context to the \*iframe \*to access all the web elements inside the iframe. Selenium WebDriver provides three ways to switch the focus to a specified iframe:

* using Index of the iframe.
* using Name or Id of the iframe
* And, using the Web Element object of the iframe.

Selenium WebDriver provides the switchTo().frame() method to switch the execution context to the identified iframe.

### How to switch to an iframe By Index in Selenium?

driver.switchTo().frame(int index);

//Switch to Frame using Index

driver.switchTo().frame(0);

//Identifying the heading in webelement

WebElement frame1Heading= driver.findElement(By.id("sampleHeading"));

//Finding the text of the heading

String frame1Text=frame1Heading.getText();

//Print the heading text

System.out.println(frame1Text);

//closing the driver

driver.close();

### How to switch to an iframe By Name or Id in Selenium?

driver.switchTo().frame("frame1");

//Switch to Frame using id of the frame driver.switchTo().frame("frame1");

//Identifying the heading in webelement

WebElement frame1Heading= driver.findElement(By.id("sampleHeading"));

//Finding the text of the heading

String frame1Text=frame1Heading.getText();

//Print the heading text

System.out.println(frame1Text);

//closing the driver

driver.close();

### How to switch to an iframe By WebElement in Selenium?

driver.switchTo().frame(WebElement iframeElement)

//Locating frame1 using its id

WebElement frame1=driver.findElement(By.id("frame1"));

//Switching the WebDriver context to frame

driver.switchTo().frame(frame1);

//Identifying the frame heading in a WebElement

WebElement frame1Heading= driver.findElement(By.id("sampleHeading"));

//Finding the text of the frame1 heading

String frame1Text=frame1Heading.getText();

//Print the heading

System.out.println("Text of the frame1 heading is:"+frame1Text);

//closing the driver

driver.close();

### How to handle nested iframes in Selenium WebDriver?

//Number of Frames on a Page

int countIframesInPage = driver.findElements(By.tagName("iframe")).size(); System.out.println("Number of Frames on a Page:" + countIframesInPage);

//Locate the frame1 on the webPage

WebElement frame1=driver.findElement(By.id("frame1"));

//Switch to Frame1

driver.switchTo().frame(frame1);

//Locate the Element inside the Frame1

WebElement frame1Element= driver.findElement(By.tagName("body"));

//Get the text for frame1 element

String frame1Text=frame1Element.getText();

System.out.println("Frame1 is :"+frame1Text);

//Number of Frames on a Frame1

int countIframesInFrame1 =driver.findElements(By.tagName("iframe")).size();

System.out.println("Number of iFrames inside the Frame1:" + countIframesInFrame1);

//switch to child frame

driver.switchTo().frame(0);

int countIframesInFrame2 =driver.findElements(By.tagName("iframe")).size();

System.out.println("Number of iFrames inside the Frame2:" + countIframesInFrame2);

driver.close();

### How to switch the context back to the parent iframe from the child iframe?

//Number of Frames on a Page

int countIframesInPage =driver. findElements(By. tagName("iframe")). size();

System.out.println("Number of Frames on a Page:"+countIframesInPage);

//Locate the frame1 on the webPage

WebElement frame1=driver.findElement(By.id("frame1"));

//Switch to Frame1

driver.switchTo().frame(frame1);

//Number of Frames on a Frame1

int countIframesInFrame1 =driver. findElements(By. tagName("iframe")). size();

System.out.println("Number of Frames inside the Frame1:"+countIframesInFrame1);

//Switch to child frame

driver.switchTo().frame(0);

int countIframesInFrame2 =driver. findElements(By. tagName("iframe")). size();

System.out.println("Number of Frames inside the Frame2:"+countIframesInFrame2);

//Locate the Element inside the Frame1

WebElement frame1Element= driver.findElement(By.tagName("body"));

//Get the text for frame1 element

String frame1Text=frame1Element.getText();

//Try to Print the text present inside parent frame

System.out.println("Frame1 is :"+frame1Text);

driver.close();

you will have to switch the context of the SeleniumWebDriver back to the parent iframe. This can be done using the method switchTo().parentFrame()

the Selenium WebDriver switch the context back to the parent iframe.

//Number of Frames on a Page

int countIframesInPage =driver.findElements(By. tagName("iframe")). size();

System.out.println("Number of Frames on a Page:"+countIframesInPage);

//Locate the frame1 on the webPage

WebElement frame1=driver.findElement(By.id("frame1"));

//Switch to Frame1

driver.switchTo().frame(frame1);

//Number of Frames on a Frame1

int countIframesInFrame1 =driver. findElements(By. tagName("iframe")). size();

System.out.println("Number of Frames inside the Frame1:"+countIframesInFrame1);

//Swiitch to child frame

driver.switchTo().frame(0);

int countIframesInFrame2 =driver. findElements(By. tagName("iframe")). size();

System.out.println("Number of Frames inside the Frame2:"+countIframesInFrame2);

//Switch to Parent iFrame

driver.switchTo().parentFrame();

//Locate the Element inside the Frame1

WebElement frame1Element= driver.findElement(By.tagName("body"));

//Get the text for frame1 element

String frame1Text=frame1Element.getText();

//Try to Print the text present inside parent frame

System.out.println("Frame1 is :"+frame1Text);

driver.close();

### How to switch the context back to the main web page from the child iframe?

In the previous section, we covered the switching between frames using Selenium. But there are scenarios when you have to get back to the main web page after working with the iframes inside the page.

Selenium WebDriver provides the method **switchTo().defaultContent().**

// Initialize browser

WebDriver driver=new ChromeDriver(); driver.get("https://demoqa.com/nestedframes");

WebElement pageHeadingElement=driver.findElement(By.xpath("//div[@class='main-header']"));

String pageHeading=pageHeadingElement.getText();

System.out.println("Page Heading is :"+pageHeading);

//Switch to Parent frame

WebElement frame1=driver.findElement(By.id("frame1"));

driver.switchTo().frame(frame1);

WebElement frame1Element= driver.findElement(By.tagName("body"));

String frame1Text=frame1Element.getText();

System.out.println("Frame1 is :"+frame1Text);

//Switch to child frame

driver.switchTo().frame(0);

WebElement frame2Element= driver.findElement(By.tagName("p"));

String frame2Text=frame2Element.getText();

System.out.println("Frame2 is :"+frame2Text);

//Switch to default content

driver.switchTo().defaultContent();

//Try to print the heading of the main page without swithcing

WebElement mainPageText=driver.findElement(By.xpath("//\*[@id='framesWrapper']/div[1]"));

System.out.println(mainPageText.getText());

driver.close();

### How to handle dynamic iframes using Selenium WebDriver?

//Find the total number of iframes on the page

int totalIFramesOnPage=driver.findElements(By.tagName("iframe")).size();

//Print the total iframes on page

System.out.println("Total iframes on Page:"+totalIFramesOnPage);

//switch to first frame using index=0

driver.switchTo().frame(0);

driver.close();

## How to handle multiple windows in Selenium?

driver.findElement(By.xpath()).click();

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

Syso(windowHandles);

Iterator <String> iterator = windowHandles.iterator();

String parentwindow = iterator.next();

Syso(parentwindow);

String childwindow = iterator.next();

Syso(childwindow);

driver.switchTo.window(childwindow);

driver.findElement(By.xpath());

driver.findElement(By.xpath());

driver.switchTo.window(parentwindow);

## Syntax For JavaScriptExecutor

JavascriptExecutor obj = (JavascriptExecutor) driver;

obj.executeScript(script, args);

### Some of the Scenarios for Using JSExecutor

Let’s discuss some of the scenarios we could use the JavaScriptExecutor Interface for Selenium.

**#1) To click the Button**

js.executeScript(“document.getElementById(‘enter element id’).click();”);

Example : JavascriptExecutor js =((JavascriptExecutor)a);

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

**#2) To send a text without using the sendkeys() method**

1. FindElement(Javascript) + Text (Javascript)

js.executeScript(“document.getElementByID(‘element id ’).value = ‘xyz’;”);

1. FindElement(WebDriver) + Text (Javascript)

WebElement details = a.findElement(By.xpath("//input[@aria-label='Details']"));

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

js.executeScript("arguments[0].value='Nitrogen details field'",details );

**#3) To generate an alert pop-up window**

js.executeScript(“alert(‘Welcome To Sotware testing help’);”);

**#4) To get InnerText of a Webpage**

string Text = js.executeScript(“return document.documentElement.innerText;”).toString();

**#5) To refresh the browser window**

js.executeScript(“location.reload()”)

**#6) To get the Title of the webpage**

String titleText = js.executeScript(“return document.title;”).toString();

System.out.println(titleText);

**#7) To scroll the webpage**

* To scroll the page vertically for 500 px:  
  js.executeScript(“window.scrollBy(0,500)”);
* To scroll the page vertically till the end:  
  js.executeScript(“window.scrollBy(0,document.body.scrollHeight)”);
* js.executeScript(“window.scrollTo(0,document.body.scrollHeight)”);

// scroll the page at the end of the page

* js1.executeScript("arguments[0].scrollIntoView();",pagescroll2); // scroll the page until the element is found
* js.executeScript("window.scrollBy(0,-500)");

// scroll the page for given index value

## We can click on the WebElement using xpath method and JavascriptExecutor method

**action.click(a.findElement(By.xpath("//input[@aria-label='Item Price']"))).sendKeys(Keys.BACK\_SPACE).build().perform(); // Click on the back space to remove the 0 number**

**We can scroll the page using JavascriptExecutor method, robot class and action class**

1. **JavascriptExecutor js =((JavascriptExecutor)a);**
2. **js1.executeScript("arguments[0].scrollIntoView();",pagescroll2);**
3. **Robot rb = new Robot();**
4. **rb.mouseWheel(100);**

**5.action.scrollToElement(doc2).perform();**

**Scroll the Option in drop-down menu and select the option**

**JavascriptExecutor js = (JavascriptExecutor) a;**

**WebElement paymentterm =a.findElement(By.xpath("//div[text()='LC']"));**

**// Locator of LC option**

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

**// Scroll the menu**

**Thread.sleep(2000);**

**paymentterm.click(); // Click on the option**

**Thread.sleep(1000);**

## Method in Java

Public class classname

{

public static void main(String[] args)

{

Classname obj = new classname();

Call the method to access it

obj.methodname(); // you can call the method multiple time

}

Syntax: public void methodname()

{

Write the selenium code

}

**}**

Here, **Public** is the access modifier

-> this method is access across different classes

**Void** is the return type

-> for return anything inside the method

-> match the data type (char, int, string) with the return type

### Static Keyword

With the use of static keyword in the method then there is no need to create obj of main main method

Public class classname

{

public static void main(String[] args)

{

**methodname();** // you can call the method multiple time

}

Syntax: public void **static**  methodname()

{

Write the selenium code

}

# Robot Class in Java

Selenium scripts use Robot class for automating the browser and desktop pop-ups,

it is part of the [Java API awt](https://docs.oracle.com/javase/10/docs/api/java/awt/Robot.html) package.

* When the user needs to handle alert pop-ups on a webpage, or
* User needs to enter text on the pop-ups with a combination of modifier keys such as Alt, Shift, etc.

We know that to perform any action on a web element, we need a locator for the element. But Windows pop-ups don't have any locators, as they are not part of the webpage, they are native OS pop-ups. To handle such pop-ups we need the Robot class.

For instance, if you are trying to download an Email Attachmeant, Windows pop-up, 'Save Attachment' prompts to specify Download Location, appears. It is nothing but a native OS pop-up.

### Keyboard methods:

* keyPress(int keycode): This method presses a given key. For Example, keyPress(KeyEvent.VK\_SHIFT) method presses ''SHIFT' key
* keyRelease(int keycode): This method releases a given key. For Example, keyRelease(KeyEvent.VK\_SHIFT) method releases ''SHIFT" key

### Mouse Methods:

* mousePress(int buttons): This method presses one or more mouse buttons.For Example, mousePress(InputEvent.BUTTON1\_DOWN\_MASK) method is used left click mouse button
* mouseRelease(int buttons): This method releases one or more mouse buttons. For Example, mouseRelease(InputEvent.BUTTON1\_DOWN\_MASK) method is used to release the left mouse button click
* mouseMove(int x, int y): This method moves the mouse pointer to given screen coordinates specified by x and y values. For Example, mouseMove(100, 50) will move the mouse pointer to the x coordinate 100 and y coordinate 50 on the screen.

## How to upload file in Selenium Webdriver ?

#### Sendkeys()

If you show type=file in the locator then use the Sendkeys method

a.findElement(By.xpath("//div[@class='dropzone dz-clickable']")).sendKeys("C:\\Users\\ameep\\OneDrive\\Documents\\Mobile testing.txt");

#### Robot class

Robot rb = new Robot(); // Create robot class to perform window events

rb.delay(2000);

// Put file path in the Clipboard such as Ctrl + C

StringSelection ss = new StringSelection("C:\\Users\\ameep\\OneDrive\\Documents\\Mobile testing.txt");

// Store the path of the file to be uploaded using StringSelection class object

Toolkit.getDefaultToolkit().getSystemClipboard().setContents(ss, null);

// Copy above path to clipboard

rb.delay(2000);

// Ctrl + V for paste the file path in the text box

rb.keyPress(KeyEvent.VK\_CONTROL);

rb.keyPress(KeyEvent.VK\_V);

rb.delay(2000);

rb.keyRelease(KeyEvent.VK\_V);

rb.keyRelease(KeyEvent.VK\_CONTROL);

rb.delay(2000);

rb.keyPress(KeyEvent.VK\_ENTER);

rb.keyRelease(KeyEvent.VK\_ENTER);

rb.delay(2000);

## Robot Class Mouse Events

* InputEvent.BUTTON1\_DOWN\_MASK : For mouse left -click
* InputEvent.BUTTON2\_DOWN\_MASK : For mouse middle button click.
* InputEvent.BUTTON3\_DOWN\_MASK : For mouse right-click
* InputEvent.BUTTON1\_MASK
* InputEvent.BUTTON2\_MASK
* InputEvent.BUTTON3\_MASK
* mouseRelease(int buttons): This method releases one or more mouse buttons. For Example, robot.mouseRelease(InputEvent. BUTTON1\_DOWN\_MASK) will release the left click press of the mouse.
* mouseMove(int x, int y): This method moves the mouse pointer to the given screen position. Here, x is X position, and y is Y position in the coordinates. For Example, the method mouseMove(100, 50) will move the mouse pointer to the x
* coordinate 100 and y coordinate 50 on the screen.

## Headless Browser Testing

[Headless Chrome](https://developers.google.com/web/updates/2017/04/headless-chrome) is a way to run the Chrome browser in a headless environment without the full browser user interface. Headless Chrome offers you a real browser context without the memory overhead of running a full version of Chrome.

Selenium WebDriver provides a class called "ChromeOptions", which can specify certain configurations to change the default behavior of Chrome. One of those configurations is the "headless" mode, which launches the Chrome in headless mode while running the test cases. The following code snippet shows how we can pass the "headless" option using the ChromeOptions class.

ChromeOptions options = new ChromeOptions()

options.addArgument("headless");

ChromeDriver driver = new ChromeDriver(options);

How can we launch the Web driver without using System set property ?

WebDriverManager

**Interview Questions & Answers**

### Typecasting

There are three type of interfaces WebDriver, JavascriptExecuter, TakeScreenshot

WebDriver driver = new ChromeDriver;

When you use ChromeDriver instead of WebDriver then you can use only ChromeDriver methods

WebDriver interface is commonly use for All the BrowserDriver

Because of these JavascriptExecutor and TakeScreenshot interface needs to typecast into WebDriver interface driver

Then you create a JavascriptExecutor reference to access JavascriptExecutor related methods

### Selenium Advantage & Disadvantage

Advantage : Support multiple language, browser and operating system

Disadvantage : not support window based application, report and data driven testing

#### 

### 

#### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 