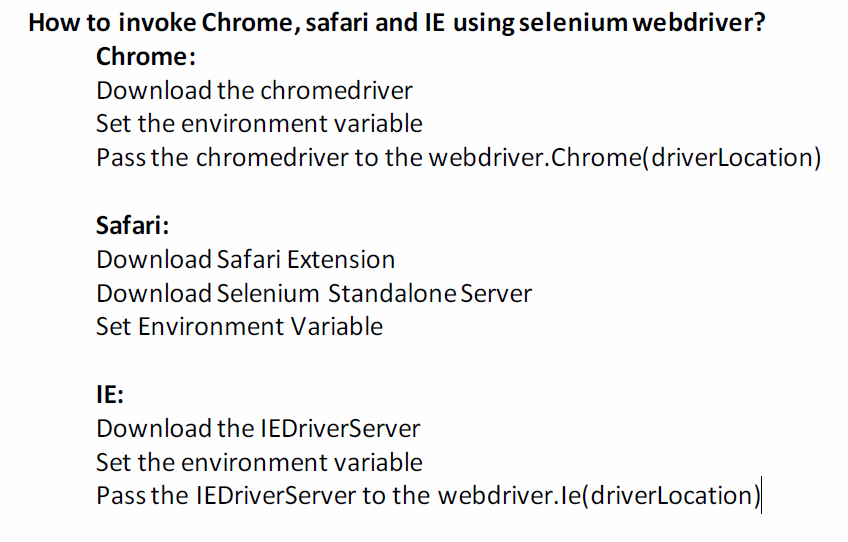
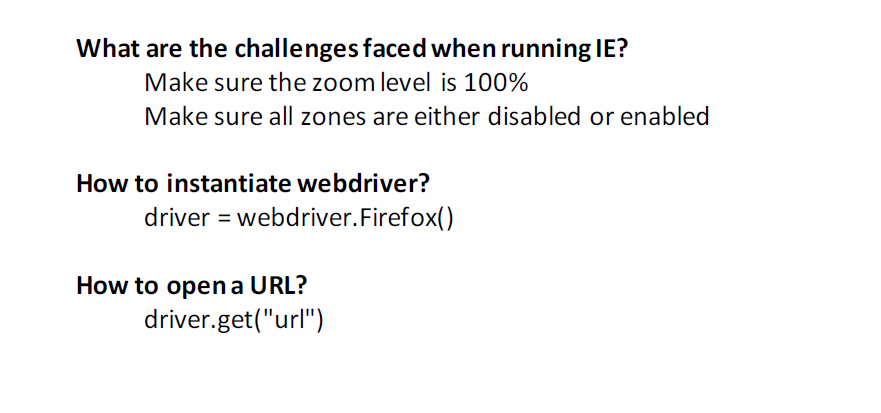
Cross Browser Testing

**Cross Browsers** testing is typically functional testing. And we expect the actual behavior in all the browsers.

Mainly the **UI/Content** will react differently in different browsers like **(IE, Firefox, Chrome and Safari)**.

Hence, we execute cross browsers testing to identify and fix





Chrome Driver

**Chrome driver** is an executable used to control Chrome browser execution on desktops and mobile devices.

Let's look into a sample script on how to set up Chrome Driver.

System.setProperty("webdriver.chrome.driver", "Local Driver Path");

driver=new ChromeDriver();

**Uses of** **Chrome driver** **include:**

* Persistent connectivity between client applications and the Chrome driver.

Mobile browser emulation for different device types

Firefox Driver

**GeckoDriver** is the latest WebDriver of Firefox browser developed by **Mozilla** foundation. From Firefox browser version 47+ onwards, we need GeckoDriver to execute the scripts.

Let's look into a sample script on how to set up Firefox Driver.

System.setProperty("webdriver.gecko.driver", "Local Driver Path");

driver= new FirefoxDriver();

**Benefits of using GeckoDriver are:**

* More secure as all browser extensions will be allowed to load only if signed by Mozilla.

Increased performance and prompt maintenance by Mozilla in case of issues.

##### InternetExplorer Driver

InternetExplorer (IE) Driver is a standalone server implemented in support for IE browser versions **7, 8, 9, 10, and 11**.

Let's look into a sample script on how to set up Internet Explorer Driver.

System.setProperty("webdriver.ie.driver","Local Driver Path");

driver = new InternetExplorerDriver();

➤ WebElement : Anything present on the webpage such as

textbox,

text,

button,

link,

table,

radio button,

checkbox etc, are called as webelement.

➤ What is the Importance of Web Element in Selenium.

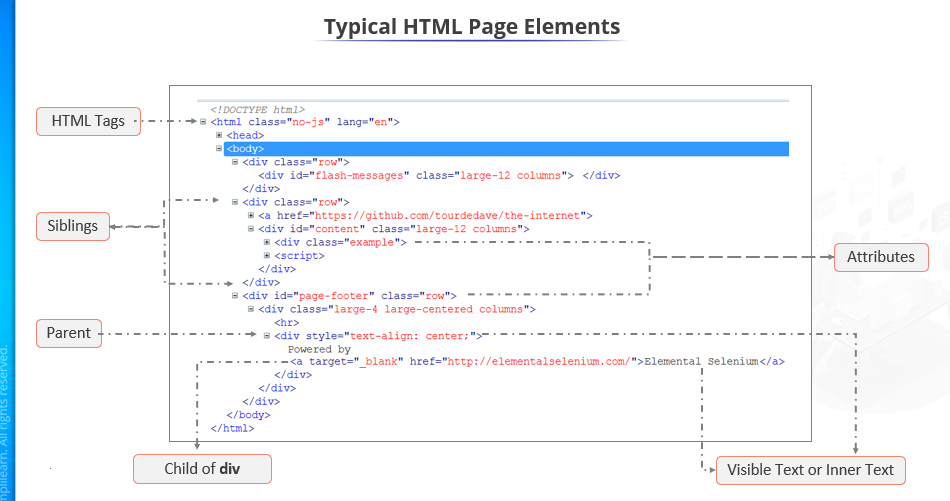
➤ The object we used to Identify and work on Web-Element is

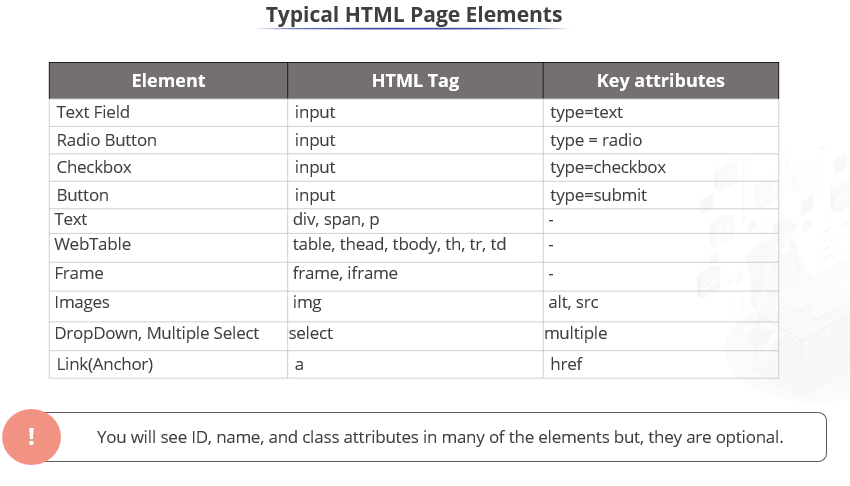
called Locator.

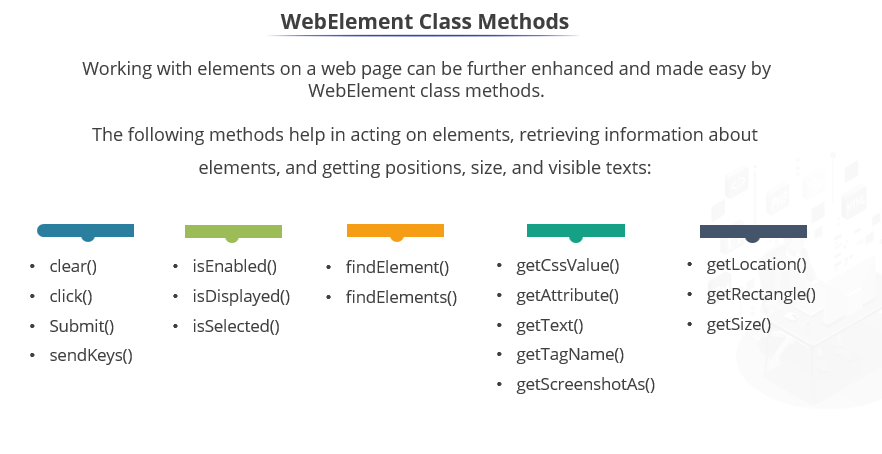
➤ Discuss WebElement Interface.

➤ WebElement can be created by findElement() and

findElements() method of WebDriver.







Locators : Selenium uses locators to find and match the elements of

the page that it needs to interact with.

➤ In Selenium we have 8 type of locators.

➤ ID

➤ Name

➤ ClassName

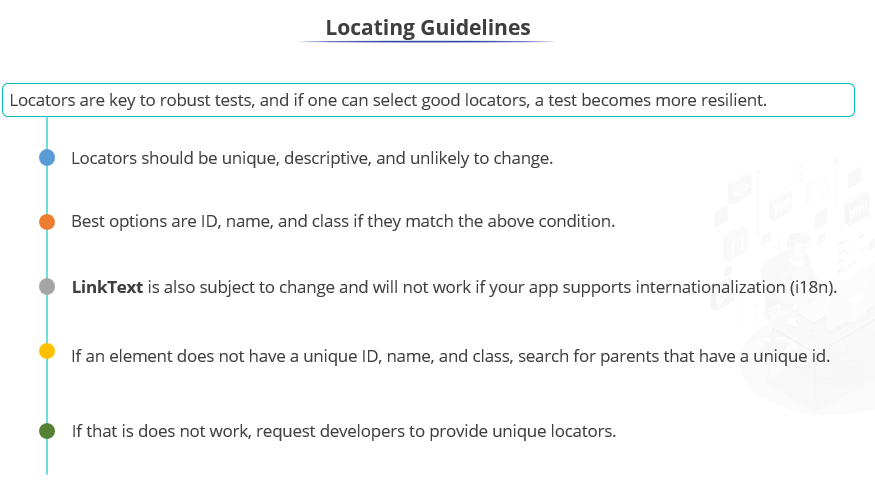
➤ TagName

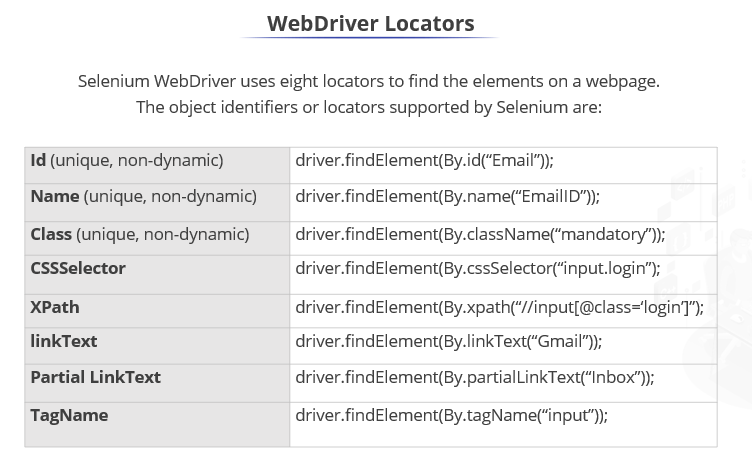
➤ Link Text

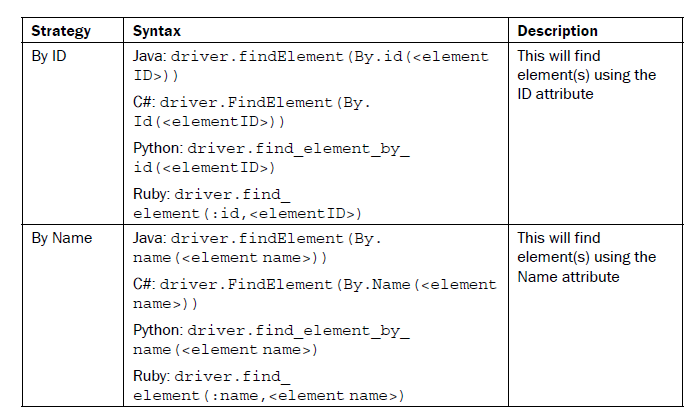
➤ Partial Link Text

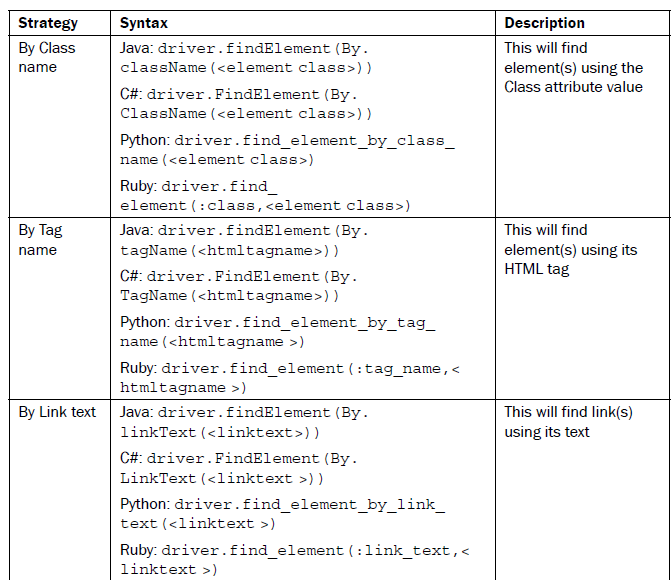
➤ CSS Selector

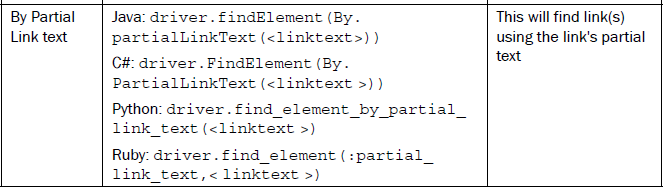
➤ XPaths

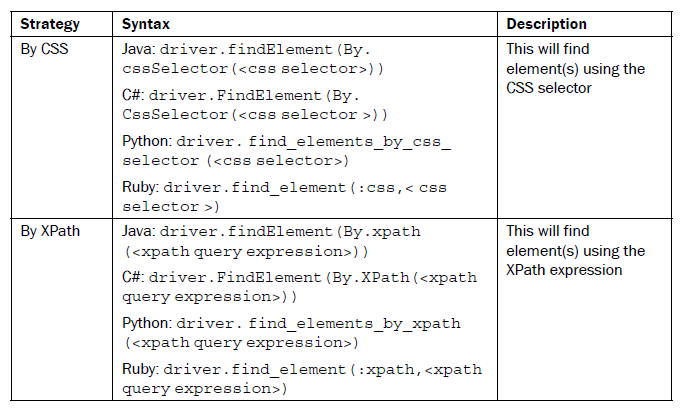












<form name="loginForm">

<label for="username">UserName: </label> <input type="text"

id="username" /><br/>

<label for="password">Password: </label> <input

type="password" id="password" /><br/>

<input name="login" type="submit" value="Login" />

</form>

To locate the User Name and Password fields, we can use the id attribute in the following way:

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

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

In this example, the login form elements use the name attribute instead of the id attribute:

<form name="loginForm">

<label for="username">UserName: </label> <input type="text"

name="username" /><br/>

<label for="password">Password: </label> <input

type="password" name="password" /><br/>

<input name="login" type="submit" value="Login" />

</form>

We can use the name attribute to find elements in the following way:

WebElement username = driver.findElement(By.name("username"));

WebElement password = driver.findElement(By.name("password"));

<form name="loginForm">

<label for="username">UserName: </label> <input type="text"

class="username" /></br>

<label for="password">Password: </label> <input

type="password" class="password" /></br>

<input name="login" type="submit" value="Login" />

</form>

We can use the class attribute to find elements in the following way:

WebElement username =

driver.findElement(By.className("username"));

WebElement password =

driver.findElement(By.className("password"));

Sometimes multiple CSS classes are given for an element. For example:

<input type="text"

class="username textfield" />

In this case, we use one of the class name with the className() method.

NoSuchElementFoundException

The findElement()methods will throw the NoSuchElementFoundException

exception when they fail to find the desired element using the specified locator strategy.

The findElements() method returns an empty list when it does not find elements

matching the locator

Finding a link by its text

WebElement gmailLink = driver.findElement(By.linkText("GMail"));

assertEquals("http://mail.google.com/",

gmailLink.getAttribute("href"));

Finding a link by partial text

WebElement inboxLink =

driver.findElement(By.partialLinkText("Inbox"));

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

Finding elements by tag name

create a test that will get all the links from a page, verify the count of links, and print a

target for each link, as follows:

@Test

public void testFindElements() {

//Get all the links displayed on Page

List<WebElement> links = driver.findElements(By.tagName("a"));

//Verify there are four links displayed on the page

assertEquals(4, links.size());

//Iterate though the list of links and print

//target for each link

for(WebElement link : links) {

System.out.println(link.getAttribute("href"));

}

}

**Finding elements using attributes with XPath**

For Relative Xpath the path starts from the middle of the

HTML DOM structure. It starts with the double forward slash

(//), which means it can search the element anywhere at the

webpage.

➤ Syntax of XPath :

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

**• // :** Select current node.

**• Tagname:** Tagname of the particular node.

**• @:** Select attribute.

**• Attribute:** Attribute name of the node.

**• Value:** Value of the attribute.

CSS Selector : CSS Selector is the combination of an element

selector and a selector value.

➤ Like Xpath, CSS selector can also locate web elements having

no ID, class or Name.

➤ CSS Selector - ID :

Syntax : <HTML tag><#><Value of ID attribute>

➤ CSS Selector - Class :

Syntax : <HTML tag><.><Value of Class attribute>

➤ CSS Selector - Attribute :

Syntax : <HTML tag><[attribute=Value of attribute]>

**Common Commands**

**Instantiating Web Elements**

Instead of using the long "driver.findElement(By.locator())" syntax every time you will access a particular element, we can instantiate a WebElement object for it. The WebElement class is contained in the "org.openqa.selenium.\*" package.

[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image016(2).png)

**Clicking on an Element**

Clicking is perhaps the most common way of interacting with web elements**. The click() method is used to simulate the clicking of any element.** The following example shows how click() was used to click on Mercury Tours'  "Sign-In" button.

[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image017(2).png)

**Get Commands**

Get commands fetch various important information about the page/element. Here are some important "get" commands you must be familiar with.

|  |  |
| --- | --- |
| **get()** *Sample usage:* | * It automatically opens a new browser window and fetches the page that you specify inside its parentheses. * It is the counterpart of Selenium IDE's "open" command. * The parameter must be a **String** object. |
| **getTitle()** *Sample usage:* | * Needs no parameters * Fetches the title of the current page * Leading and trailing white spaces are trimmed * Returns a null string if the page has no title |
| **getPageSource()** *Sample usage:* | * Needs no parameters * Returns the **source code of the page** as a String value |
| **getCurrentUrl()** *Sample usage:* | * Needs no parameters * Fetches the string representing the **current URL** that the browser is looking at |
| **getText()** *Sample usage:* | * Fetches the **inner text** of the element that you specify |

**Navigate commands**

These commands allow you to  refresh,go-into and switch back and forth between different web pages.

|  |  |
| --- | --- |
| **navigate().to()** *Sample usage:* | * It automatically **opens a new browser window and fetches the page** that you specify inside its parentheses. * **It does exactly the same thing as the get() method.** |
| **navigate().refresh()***Sample usage:* | * Needs no parameters. * It **refreshes** the current page. |
| **navigate().back()***Sample usage:* | * Needs no parameters * Takes you **back by one page** on the browser's history. |
| **navigate().forward()***Sample usage:* | * Needs no parameters * Takes you **forward by one page** on the browser's history. |

**Closing and Quitting Browser Windows**

|  |  |
| --- | --- |
| **close()** *Sample usage:* | * Needs no parameters * **It closes only the browser window that WebDriver is currently controlling**. |
| **quit()** *Sample usage:* | * Needs no parameters * **It closes all windows that WebDriver has opened.** |

Taking ScreenShot

The screenshot helps to check the functionality, debug and find out the issue, and identify once the test cases fail.

**Steps to take the screenshot**

* It helps to convert web driver object to TakeScreenShot.

TakesScreenShot scrShot =((TakesScreenShot)webdriver);

* It calls getScreenShotAs method to create an image file.

File SrcFile=scrShot.getScreenShotAs(OutputType.FILE);

* Now we are copying file to a desired location.

FileUtils.copyFile(source, new File("D:/selenium\_workspace/error.png")).

There are two kinds of waits.

1. Implicit wait - used to set the default waiting time throughout the program
2. Explicit wait - used to set the waiting time for a particular instance only

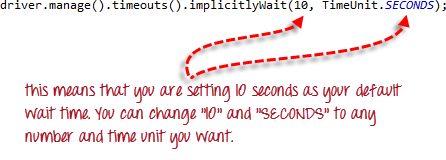
**Implicit Wait**

* It is simpler to code than Explicit Waits.
* It is usually declared in the instantiation part of the code.
* You will only need one additional package to import.

To start using an implicit wait, you would have to import this package into your code.

[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image038(1).png)

Then on the instantiation part of your code, add this.

[](https://www.guru99.com/images/image039(1).png)

**Explicit Wait**

**Explicit waits are done using the WebDriverWait and ExpectedCondition classes**. For the following example, we shall wait up to 10 seconds for an element whose id is "username" to become visible before proceeding to the next command. Here are the steps.

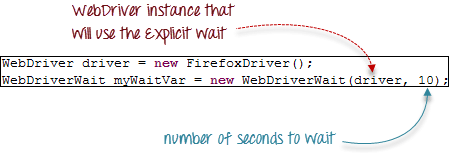
**Step 1**

Import these two packages:

**[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image040(1).png)**

**Step 2**

Declare a WebDriverWait variable. In this example, we will use "myWaitVar" as the name of the variable.

[](https://www.guru99.com/images/image041(1).png)

**Step 3**

Use myWaitVar with ExpectedConditions on portions where you need the explicit wait to occur. In this case, we will use explicit wait on the "username" (Mercury Tours HomePage) input before we type the text "tutorial" onto it.

[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image042(1).png)

**These are the available expected conditions:**

* alert\_is\_present
* element\_located\_selection\_state\_to\_be
* element\_located\_to\_be\_selected
* element\_selection\_state\_to\_be
* element\_to\_be\_clickable
* element\_to\_be\_selected
* frame\_to\_be\_available\_and\_switch\_to\_it
* invisibility\_of\_element\_located
* presence\_of\_all\_elements\_located
* presence\_of\_element\_located
* staleness\_of
* text\_to\_be\_present\_in\_element
* text\_to\_be\_present\_in\_element\_value
* title\_contains
* title\_is
* visibility\_of
* visibility\_of\_element\_located

**Web-Based Popups**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/webdriver-alerts-1.jpg)

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

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

driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click();

Thread.sleep(**5000**);

// accepting javascript alert

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

alert.accept();

// clicking on try it button

driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click();

Thread.sleep(**5000**);

// accepting javascript alert

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

// clicking on try it button

driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click();

Thread.sleep(**5000**);

// accepting javascript alert

System.out.println(driver.switchTo().alert().getText());

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

**Window popup/Windows Handling:**

package Selenium;

import java.util.Iterator;

import java.util.Set;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class WindowPopup {

public static void main(String[] args) throws InterruptedException {

System.setProperty("webdriver.chrome.driver", "D:/aaa/Chrome driver/chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://demo.com/popup.php ");

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

Thread.sleep(2000);

driver.findElement(By.linkText("Click Here")).click();

Thread.sleep(2000);

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

Iterator < String > it = s.iterator();

it.next();

String w2nd = it.next();

driver.switchTo().window(w2nd);

Thread.sleep(2000);

driver.findElement(By.name("emailid")).sendKeys("gaurav.3n@gmail.com");

Thread.sleep(2000);

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

Thread.sleep(2000);

driver.findElement(By.linkText("Click Here")).click();

}

}

Web Table

Let's create a simple test that will print data from a table, as shown in the following code

example:

@Test

public void testWebTable() {

WebElement simpleTable = driver.findElement(By.id("items"));

// Get all rows

simpleTable.findElements(By.tagName("tr"));

assertEquals(3, rows.size());

// Print data from each row

for (WebElement row : rows) {

List<WebElement> cols = row.findElements(By.tagName("td"));

for (WebElement col : cols) {

System.out.print(col.getText() + "\t");

}

System.out.println();

}

}

Actions Class :

baseurl = **"https://jqueryui.com/droppable/"**

driverLocation = **"C:\\Softwares\\chromedriver.exe"**baseurl = **"https://jqueryui.com/droppable/"**driver = webdriver.Chrome(driverLocation)  
driver.get(baseurl)  
driver.maximize\_window()  
driver.implicitly\_wait(5)  
driver.switch\_to.frame(0)  
fromelement=driver.find\_element(By.ID,**"draggable"**)  
toelement=driver.find\_element(By.ID,**"droppable"**)  
time.sleep(2)  
**try**:  
 action =ActionChains(driver)  
 action.drag\_and\_drop(fromelement,toelement).perform()  
 time.sleep(3)  
 print(**"Drag and drop successfull"**)  
 driver.close()  
**except**:  
 print(**"Drag and drop not successfull"**)

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;

public class ActionsContextClickDemo {

public static void main(String[] args) throws InterruptedException {

System.setProperty("webdriver.chrome.driver", "D:/aaa/Chrome driver/chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

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

Thread.sleep(2000);

WebElement a = driver.findElement(By.linkText("Gmail"));

Actions b = new Actions(driver);

b.contextClick(a).build().perform();

}

}-----------------------------------------------------------------

5)perform()

This method is used for perform action on particular web element

**Page object Model(POM):**

In normal java programing constructor are mainly used to initialize data member or variable

public class Test

{ int a ; // variable declaration

}

Test()

{

a=20; //initialization

}

public void test1()

{

s.o.p(a); //use

}

**Concepts use**

1)Encapsulation

2)Annotation

**1)Encapsulation:**

Whenever in oops ,we have to make any data member of class usable for only that class ,that time we declare it as private, this is known as Encapsulation .

OR

Encapsulation is the Wrapping of the data .

---------------------------------------------------

**2)Annotation**

Annotation contains some code whenever we use annotation, then at time of execution that code is get executed.

@findBy(xpath="xpathExpression")

**POM Class**

1)POM class1

Example:

package pom\_class;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class PomDemo1 {

@FindBy(xpath="//input[@name='q']") private WebElement SEARCH;

public PomDemo1(WebDriver driver)

{

PageFactory.initElements(driver,this) ;

}

public void search() {

SEARCH.sendKeys("Selenium");

}

}

------------------------------------------------------------

2)POM class2

Example:

package pom\_class;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class PomDemo2 {

@FindBy(xpath="//a[@class='gb\_f'][1]") private WebElement GMAIL;

public PomDemo2(WebDriver driver)

{

PageFactory.initElements(driver,this) ;

}

public void gmail() {

GMAIL.click();

}

}

3.Main Class:

package pom\_class;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.chrome.ChromeDriver;

public class TestClass {

public static void main(String[] args) throws InterruptedException {

System.setProperty("webdriver.chrome.driver", "E:/Soft/chromedriver.exe"); WebDriver driver=new ChromeDriver(); driver.get("https://www.google.co.in/"); driver.manage().window().maximize();

Thread.sleep(2000);

PomDemo1 pom = new PomDemo1(driver); pom.search(); Thread.sleep(2000);

PomDemo2 pom1 = new PomDemo2(driver); pom1.gmail();

}

}