SELENIUM

Contents

Disadvantages of Manual Testing

- Time consuming
- Requires more manpower
- Tedious and monotonous job
- It is susceptible to human errors

What is Automation?

Testing a software with the help of any tool is called as automation testing

Advantages of Automation

- It is faster
- Saves time
- Reduces human effort
- More accurate

Disadvantages of Automation

- Initial Investment is high
- Required skilled manpower

What is Selenium?

Selenium is an automation tool, which is used to automate web based applications.

It is a collection of libraries which consists of inbuilt classes and methods.

Advantages of Selenium

- Open source.
- It supports 14+ programming languages.
- It supports all major browsers(e.g. Chrome, Firefox ,Edge , IE,Opera….)
- It supports all major Operating Systems(e.g. Windows , Linux , mac)
- It allows 3rd party integration(e.g. TestNg , AutoIT, Cucumber…)

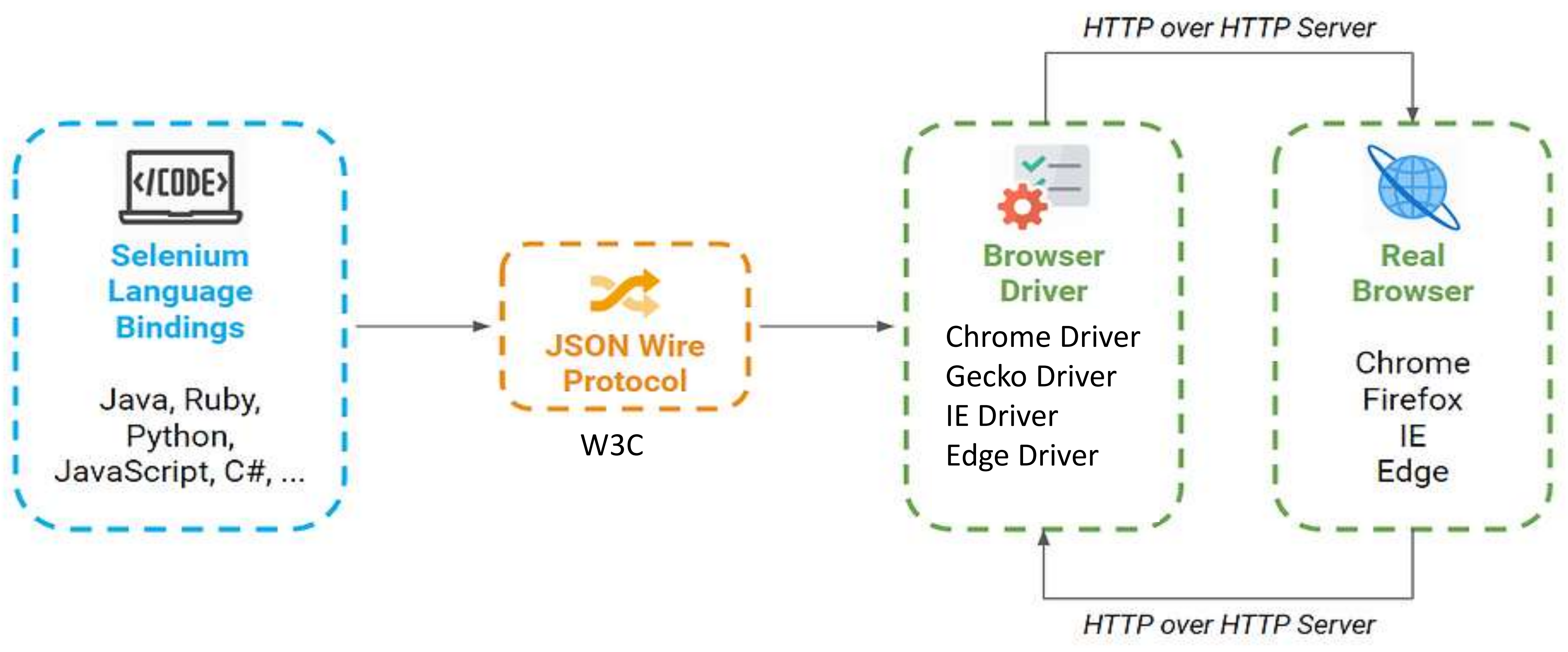
Disadvantages of Selenium

It Supports only web based applications.

No customer support.

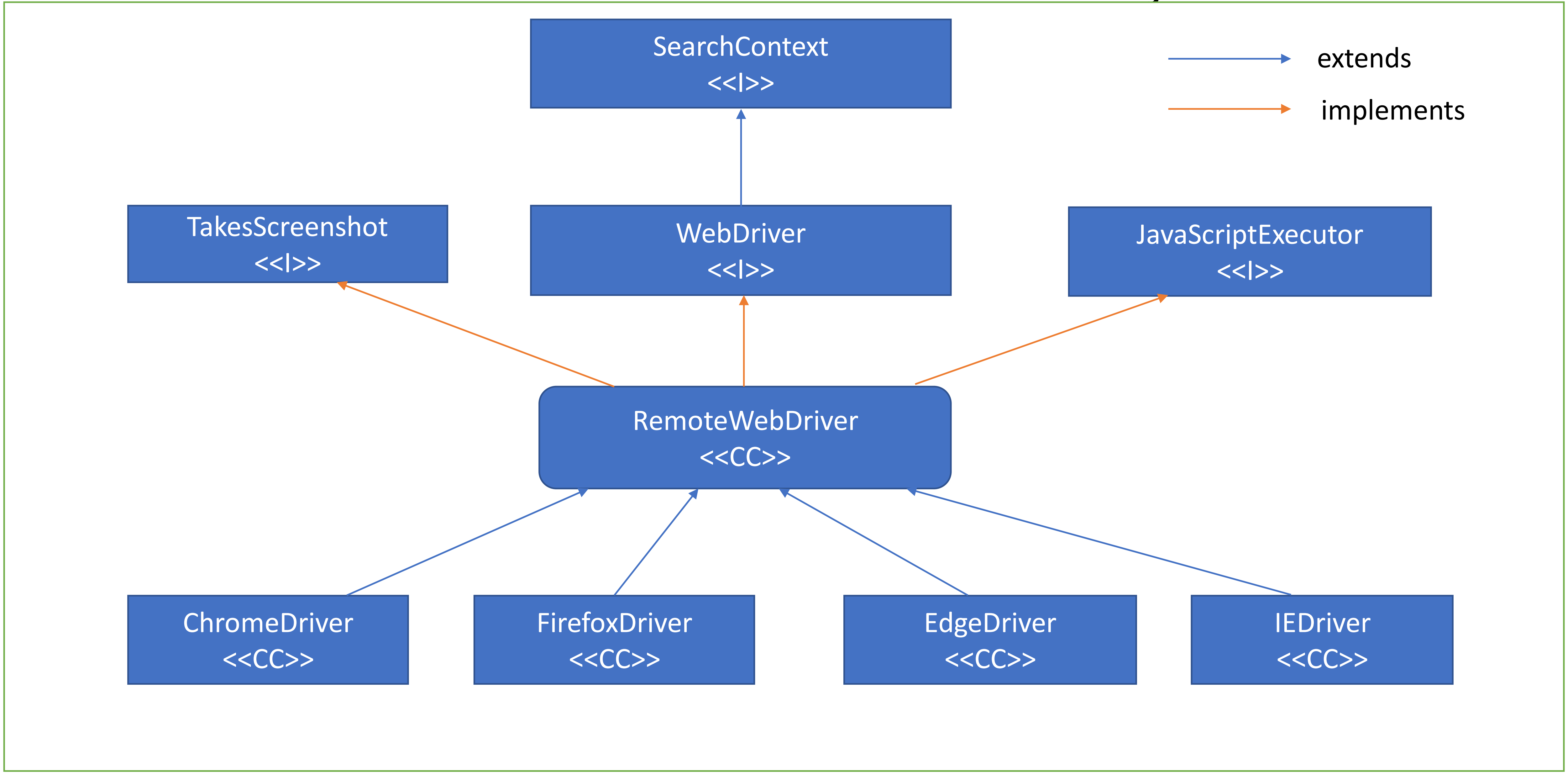
We can not automate Captcha , OTP and animations.

Selenium WebDriver Architecture



- Script Written in any programming language should be bind with selenium this step is called as language binding/client binding.
- Since browser understands only http request ,The script is transferred through JSON(W3C) wire protocol which will convert our script into browser understandable language i.e http request.
- Each browser vendors provide their own drivers to communicate with browser.
- When a browser driver receives request, that will be send to browsers in the form of http request.
- Browser performs specified action and it will send back the response in the form of http response.
- So we can call this process as bidirectional process.

Selenium WebDriver Architecture w.r.t java



SearchContext

- It provides a mechanism to search web element in webpage.
- It provides 2 abstract methods
  1. findElement(By by)
  2. findElements(By by)

WebDriver

- It provides mechanism to perform browser related actions.
- It provides 11 abstract methods(direct) and 2 inherited methods.

TakesScreenshot

- It provides mechanism to take a screenshot of entire web page.
- It provides 1 abstract method
  1. getScreenshotAs()

JavaScritExecutor

- It provides a mechanism to write a script so that browser understands(JS).
- It provides 2 abstract methods.
  1. executeScript()
  2. executeAsyncScript()

Note: All the implementation of these methods is given in RemoteWebDriver class.
Respective browser specified classes extends RemoteWebDriver class.

Installation steps

- Create a new java project in eclipse
- Right click on java project and create 2 folders , name it as jars and drivers

Steps to install jars and drivers:

- ➢ Search selenium.dev in chrome.
- ➢ Click on downloads in top nav bar.
- ➢ To download jar file:
  - Find Previous releases and click on releases.
  - Click on selenium version 3.141.59
  - Click on selenium-server-standalone-3.141.59jar
- ➢ To download driver file
  - In the same page click on browsers.
  - Click on documentation of respective browser.
  - Click on respective browser version.
  - Download zip file.(eg: chromedriver_win32.zip)
  - ➢ Extract the zip file
- ➢ Copy the jar file to jars folder in eclipse and add to build path.
- ➢ Copy driver.exe file to drivers folder in eclipse.

To launch an empty browser

- Step1:

Set the property of a driver executable file.

We have a method called setProperty(String key,String value) which is present in System class.

System-->It is an inbuilt final class present in java.lang package.

setProperty-->It is a static method present in System class

It takes 2 arguments(key and value in form of String)

Key =Specifies which browser we want to launch(given by brower vendors).

Value=Specifies path of driver executable file.

- Step 2:

Create an instance of browser specific class.

Example:

```
 public  class ToLaunchAnEmptyBrowser {
          public static void main(String[] args) {
          System.setProperty(“webdriver.chrome.driver”, “./drivers/chromedriver.exe”);
          ChromeDriver driver=new ChromeDriver();
          }
}
```

WebDriver Methods

1. **get(String url)**
    - It is used to navigate to an application.
    - URL should be fully qualified path, otherwise we will get “InvalidArgumentException”.
    - Return type is void.
2. **getTitle()**
    - It is used to capture the title of the page.
    - return type is String.
3. **getWindowHandle()**
    - It is used to capture the window id of a webpage, where the driver control is present.
    - return type is String.
4. **getWindowHandles()**
    - It is used to capture all the window ids of a webpage that are opened from selenium.
    - return type is Set<String>.
5. **close()**
    - It is used to close the browser where the driver control is present(focused page).
    - It won’t stop the server.
    - return type is void.
6. **quit()**
    - It is used to close all browser windows that are opened by selenium.
    - It will stop the server as well.
    - return type is void.

7. **getCurrentUrl()**

- It is used to capture the url of the webpage.
- Return type is String.

8. **getPageSource()**

- It is used to capture the source code(HTML,CSS,JavaScript) of the webpage.
- Return type is String

9. **manage()**--> Return type is Options(I) type object ref

It is used for managing a) window()

b) timeouts()

c) cookies()

a)window()--> Return type is Window(I) type object ref

- fullScreen()-->used to make browser fullscreen
- maximize()-->used to maximize the browser window
- setSize(Dimension)-->to set size of the browser
- getSize()-->to get height and width of the webpage
- setPosition(Point)-->to set the cursor position
- getPosition()-->to get the cursor position

10. **navigate()**

- It is used to navigate within the application.
- Return type is Navigation type object.

a. forward()

b. back()

c. refresh()

$$\left.\begin{array}{l}\text{d. to (String arg0)}\\ \text{e. to (URL arg0)}\end{array}\right\}\ \text{Internally calls get()}$$

11. **switchTo()**

- It is used to switch driver control to
  a) window()
  b) frame()
  c) alert()
- Return type is TargetLocator type object.

SearchContext Methods:

1. **findElement(By by)**

- It helps us to locate one element at a time.
- It takes an argument called By, where we should pass By type object ref.
- Return type is WebElement.

2. **findElements(By by)**

- It helps us to locate multiple elements at a time.
- It takes an argument called By, where we should pass By type object ref.
- Return type is List<WebElement>.

By:

- By is an abstract class.
- It provides 8 static methods
  1. id()
  2. name()
  3. className()
  4. linkText()
  5. partialLinkText()
  6. tagName()
  7. cssSelector()
  8. xpath()
- All these methods accepts “String” as an argument.
- Return type is By type Object ref.

Understanding on HTML

What is HTML?

HTML:-Hyper Text Markup Language

- It is used to develop a web page.
- It is made up of tags.

Core Structure of HTML

```
<html>
        <head>
                <title>…..</title>
        </head>
        <body>
        </body>
</html>
```

Tags:

1. Paired tag
2. Unpaired tag/self closing tag

Tagname:

The very first word after "<" symbol, until space is called as tagname.

ex: <div …..> </div>

Paired tag:

- Tag which has both opening and closing tags is considered as Paired tag.
- They can have child tags , plain text and attributes.

Ex: <p></p>, <h1></h1>, <body></body>   etc.

Unpaired tag:

- Unpaired tags are opened tag and do not have to be closed.
- They are also called as self closing tags.

Ex: <br>,<img> etc.

Attributes:

- It provides additional information about the tag.
- Any key value pair inside opening tag is considered as attribute.

o attribute name:- key of an attribute.

o attribute value:- value of an attribute.

Ex: <input type=“text” id=“username”> </input>

Paths

- **Absolute path:**

Starting from the html tag to the desired element or the required element is called an absolute path.

Absolute path starts with / (single forward slash)

Syntax :- ./html/body/tag_name/child tag_name

- **Relative Path:**

Relative path starts with // (double forward slash) which represents descendants (child, grandchild, great grandchild…).

Syntax:- //tag_name/child tag_name

Note :-

/ => represents immediate child

// => represents descendant

Locators

- It is used to identify the web elements in the web page.
- Types of Locators

$$\left.\begin{array}{l}\text{1) id}\\ \text{2) name}\\ \text{3) linkText}\\ \text{4) partialLinkText}\\ \text{5) className}\\ \text{6) tagName}\end{array}\right\}\text{ Direct locators}$$

$$\left.\begin{array}{l}\text{7) CSS selector}\\ \text{8) X-path}\end{array}\right\}\text{ Expressions}$$

Note :- All these methods are present in “By class” and they are static methods which will take “String” as an argument.

Ex: By.id(“ ”)

By.name(“ “)

1. **id() :**

- It is used to identify an element using “id” attribute present in html code.
- It is given topmost priority.

2. **name()**

- It is used to identify an element using “name” attribute present in html code.
- It is given 2nd priority.

3. **linkText()**

- It is used to identify an element using text which is present between anchor tag (<a>text</a>)
- It is given 3rd priority.

4. **partialLinkText()**

- we go for partial link text only when,

✓ link is very lengthy.

✓ link is partially dynamic.

✓ when we have blank spaces at the beginning or at the end of a text.

5. **className()**

- It is used to identify an element using “class” attribute present in html code.
- It is least recommended, because it is dynamic in nature.

6. **tagName()**

- It is used to identify an element using tagname.
- It is not recommended, because of multiple elements will be developed using same tag name.

7. **cssSelector()**

- It is used to identify an element using following syntax.

  tagname[Attribute name=“Attribute value”]
- Tag must contain atleast one attribute
- It is unidirectional.

8. **xpath()**

- It is one of the relative path.
- It is multidirectional.

Syntax:

a) xpath by attribute :- //tagname[@Attribute name=‘Attribute value’]

b) xpath by text function :- //tagname[text()=‘text value’]

c) xpath by contains :- //tagname[contains(@AN,‘AV’)] (or) //tagname[contains(text(),’TV’)]

d) xpath by multiple attributes :- //tagname[@Attribute name=‘Attribute value’ and @Attribute name=‘Attribute value’] (or)
//tagname [@Attribute name=‘Attribute value’ or text()=‘TV’]

e)Dependent and independent xpath

Steps:

1) Identify dependent and independent element.
2) Write xpath for independent element.
3) Traverse back to meet immediate common parent(/..).
4) Update xpath expression by writing xpath for dependent element.

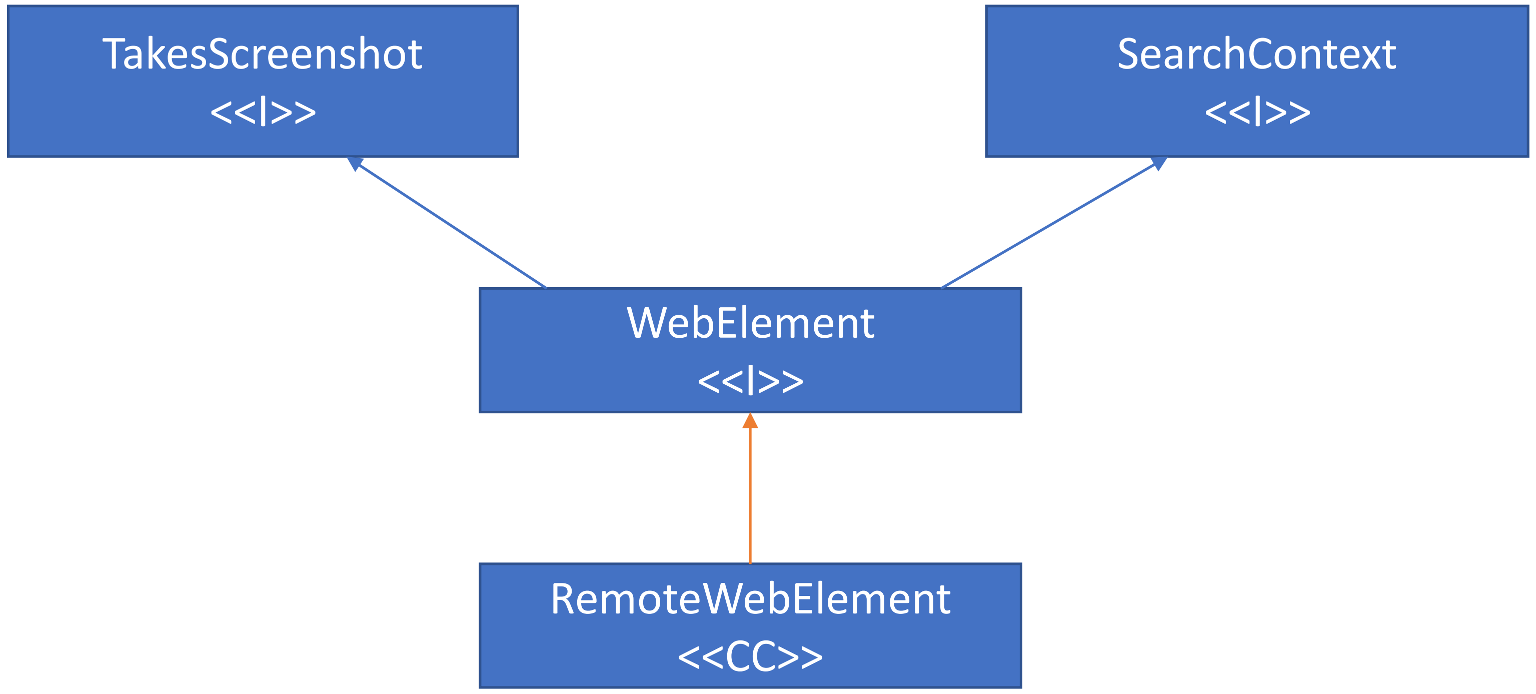
Ex: Add to cart , check box ,radio button…..etc.

f)Xpath by Axes

Syntax:- //tagname[@AN='AV']/axisname::tagname

axisname => following-sibling (or) preceding-sibling (or) ancestor

WebElement



TakesScreenshot:
It has a method called getScreenshotAs() used to take the screenshot of a web element
SearchContext:
It has 2 methods a)findElement(By by) b)findElements(By by)
WebElement:
It has 14 inbuilt methods and 3 inherited methods.
RemoteWebElement:
All the above interface methods implementation will be given in this class.

WebElement methods:

1. click()
2. submit()
3. sendKeys(CharSequence…)
4. clear()
5. getText()
6. getAttribute(String)
7. getTagname()
8. getCssValue(String)
9. getSize()
10. getLocation()
11. getRect()
12. isDisplayed()
13. isSelected()
14. isEnabled()

1. **click()**

- It is used to perform click action on an element.
- Return type is void.

2. **submit()**

- It is also used to perform click action on an element.
- Return type is void.
- It must satisfy the following conditions:
  1. type must be “submit”.
  2. “child of form” tag.

3. **sendKeys(CharSequence…)**

- It is used to enter the data inside text field.
- It accepts enum as an argument.
- We have inbuilt enum called Keys.
- Inside Keys we have all the keyboard buttons.

4. **clear()**

- It is used to clear the content which is entered inside text field.
- Return type is void.

5. **getText()**

- It is used to capture the text of an element.
- Return type is String.
- It is used for validation purpose.

6. **getAttribute(String)**

- It is used to capture the value of an attribute.
- Return type is String.

7**. getTagName()**

- It is used to capture the tag name of an element.

8. **getCssValue(String)**

- It is used to capture the CSS value (Ex: color, font-size , font-family… etc)
- Return type is String.

9. **getSize()**

- It is used to capture the size of an element.
- It returns height and width of the element in pixel.

10. **getLocation()**

- It is used to capture the x and y coordinates in pixels.

11. **getRect()**

- It is used to capture height ,width, x location , y location.
- It returns Rectangle type object.

12. **isDisplayed()**

- To check whether the web element is displayed or not.
- Return type is boolean.

13. **isSelected()**

- To check whether the web element is selected or not.
- Applicable if the element is developed using select tag.
- Return type is boolean.

14. **isEnabled()**

- To check whether the web element is enabled or not.
- To use this button tag should contain “disabled” attribute.

TakesScreenshot

- It is an interface which provides the mechanism to take screenshot.
- It has two different implementation

    1. RemoteWebDriver (Entire web page)
    2. RemoteWebElement (of a web element)

- It provides 1 abstract method → **getScreenshotAs( )**
- Return type is based on argument passed.

  Example :- getScreenshotAs(OutputType.FILE)

  getScreenshotAs(OutputType.BYTES)

  getScreenshotAs(OutputType.BASE64)

Note:

- It is a good practice to store the screenshot inside a folder called as screenshots/errorShots.
- Whenever there is a defect it is a best practice to add the screenshot is defect report.
- Whenever any automation script fails we take a screenshot.

How to take a screenshot

- getScreenshotAs() will take the screenshot and store it in a temporary directory and this screenshot will be deleted after the program ends.
- So we should move the file to a different location before the program ends.
- To move the file to a different location we use copy() from FileHandler class.

Ex1:- <u>To take a screenshot of an entire web page</u>

```
TakesScreenshot  ts = (TakesScreenshot) driver;
File temp = ts.getScreenshotAs(OutputType.File);
File dest= new File(“./errorShots/error1.png”);
FileHandler.copy(temp,dest);
```

Ex2:- <u>To take a screenshot of a web element</u>

```
File temp = driver.findElement(By by).getScreenshotAs(OutputType.File);
File dest= new File(“./errorShots/error2.png);
FileHandler.copy(temp,dest);
```

Synchronization/WebDriver Wait

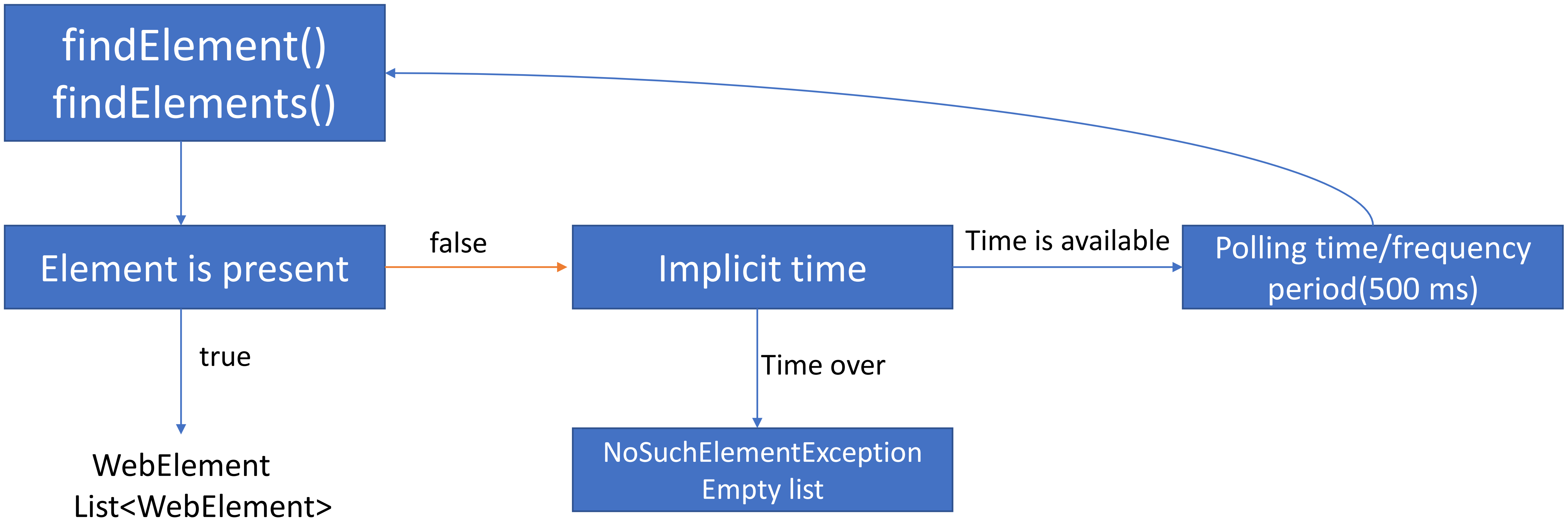
- The process of matching the selenium speed with the application speed is called synchronization.
- When the page loads on a browser, various web elements on it with may load at different time intervals.
- So we make use of wait commands to direct a test script to pause for a certain time before throwing Exception.
- We go for wait commands to reduce the usage of Thread.sleep()

[blind wait/dead wait]

Types of wait commands

1. Implicit wait
2. Explicit wait
3. Fluent wait

Implicit Wait



Syntax :-

```
driver.manage().timeouts().implicitlyWait(time, TimeUnit.SECONDS);
```

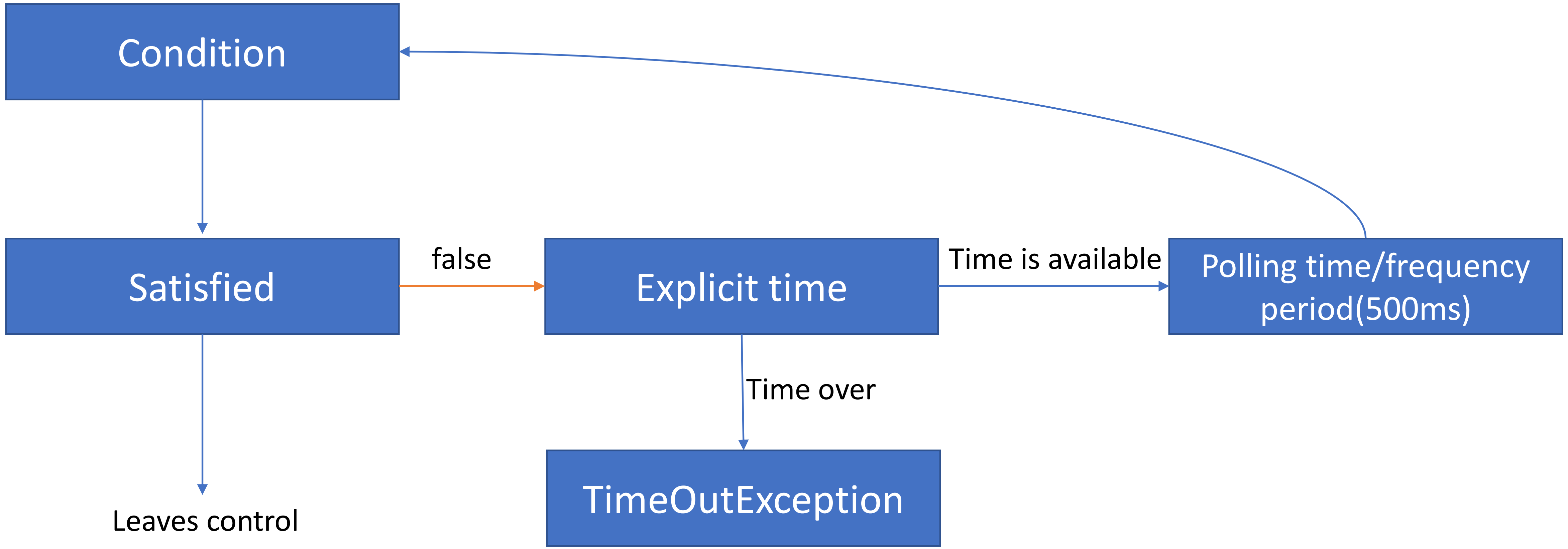
It takes 2 arguments

1. time duration(long)
2. TimeUnit(microseconds, milliseconds,seconds,minutes,hours,days….etc)

- Implicit wait will work only for findElement and findElements.
- When the control comes to any findElement or findElements statement it will check whether the element is present or not.
- If the element is present then the findElement will return the first matching element, where as findElements returns all the matching elements.
- If the element is not present then it check for implicit time provided.
- If the time is available it will wait for 500ms which is called the polling period,

  then it will continue to check whether the element is present or not
- If the time is over then findElement throws NoSuchElementException, where as findElements returns empty list.

Explicit Wait



Syntax :-

```
      WebDriverWait  wait = new WebDriverWait( driver, Seconds);
       wait.until(ExpectedConditions.titles(“ actiTime – Enter Time-Track”));
```

- In order to handle the synchronization of any method, we can use explicit time .
- WebDriverWait itself is called an explicit wait, because we have to specify the waiting condition explicitly.
- When the control comes to wait.until statement it will check the specific condition.
- If the condition is true it will go to the next statement, if the condition is false it will check for explicit time provided.
- If the time is over, it will throw TimeOutException, else it will wait for 500ms and it will continue to check the condition.

Note :

Expectedcondition is an interface and ExpectedConditions is a class.

Fluent Wait

- It is same as Explicit wait.
- It is used to define our own polling time.
- It is not widely used in industry

Difference between Implicit and Explicit wait

Implicit wait	Explicit wait
It works for findElement() and findElements()	It works only for the given condition
We can provide different time units(sec, milli sec,minutes,hours,days…)	We can provide time unit only in seconds
If the element is not present we get NoSuchElementException or empty list	If the element is not present we get TimeOutExcpetion

Note:-

Synchronization is the main challenge faced by automation engineers.

Select Class (Dropdown/listbox)

- There are two types of listbox
  1. Single select listbox
  2. Multi select listbox
- Dropdown will be created using “select” tag and content of the dropdown will be created using “option” tag.
- To handle dropdown we use the Select class of selenium.
- Select class has parameterized constructor, where it takes an argument of type WebElement (address of the dropdown)

Syntax :- Select dropdown = new Select(WebElement address);

- In order to perform action on the dropdown we can use any one of the following methods.

1. selectByIndex(int)
2. selectByValue(String)
3. selectByVisibleText(String)
4. deSelectByIndex(int)
5. deSelectByValue(String)
6. deSelectByVisibleText(String)
7. deSelectAll()
8. getFirstSelectedOption()
9. getAllSelectedOption()
10. isMultiple()
11. getOptions()

iFrame

- A web page inside another web page is called as embedded web page(frames).
- Embedded web pages are created using the “iframe” tag.

Purpose of iframe :

- Advertisement.
- Commonly used features (captcha).
- Common page (If two or more developers working on a same page).

Syntax :-

```
driver.switchTo().frame()
```

frame() is overloaded, We can switch our driver control from parent window to frame in three ways

1. frame(int index)
2. frame(String name/id)
3. frame(WebElement ele)

Note :-

- To switch back the driver control we have two methods

1. driver.switchTo().parentFrame() ---->To switch back the driver control to immediate parent frame.
2. driver.switchTo().defaultContent()----->To switch back the driver control to main page.

Actions Class

- Actions class is mainly used for mouse actions.
- By using mouse actions we can perform following actions

  1) Handling mouse hover (dropdown menu → moveToElement( ))
  2) Right click
  3) Drag and drop
  4) Double click

- Actions class has a parameterized constructor where it takes WebDriver as an argument

Syntax :-

```
Actions action = new Actions (driver);
```

- Whenever we call any methods of actions class we have to call perform ( ) at the end

Methods of Actions class

1) perform()
2) click()
3) click (WebElement)
4) doubleClick()
5) doubleClick (WebElement)
6) contextClick()
7) contextClick (WebElement)
8) moveToElement (WebElement)
9) moveToElement (WebElement , int xoff , int yoff)
10) moveByOffSet (int xoff ,int yoff)
11) dragAndDrop (WebElement source , WebElement target)
12) dragAndDragBy (WebElement source , int xoff , int yoff)

13) clickAndHold( )

14) clickAndHold(WebElement ele)

15) release( )

16) release(WebElement ele)

17) sendKeys (WebElement , Charseq…ch)

18) sendKeys (Charseq…ch)

19) KeyUp (Charseq…ch)

20) KeyDown (Charseq…ch)

21) build( )

Note :- Now build () is internally present in perform ()

PopUps

- PopUps are GUI which appears on window when end-users perform some action on web element.

<u>Purpose of PopUps :-</u>

I. To grab attention of end-users.

II. To give information to end-users.

III. To collect information from end-users.

<u>PopUps are categorized into following types :-</u>

1. Javascript popup
2. Hidden division popup/modals/overlays
3. File upload popup
4. File download popup
5. Notification popup
6. Authentication popup
7. Child window popup

1. Javascript popups

- These popups are developed using javascript language.
- We can not avoid these popups.

<u>Types of Javascript popups</u>

1. Alert popup
2. Confirmation popup
3. Prompt popup

<u>Characteristics of javascript popups</u>

1. We cannot inspect this popup.
2. We cannot move this popup.
3. We cannot perform any action on the webpage unless you handle the popup.
4. This popup will have OK button(alert) and if it contains OK and Cancel button(Confirmation) and if it contains text field (Prompt).

To Handle javascript popup

Syntax :- driver.switchTo().alert()

1. accept() :- To click on the OK button.
2. dismiss() :- To click on the cancel button.
3. getText() :- To get the text present on the popup.
4. sendKeys() :- To enter the text on the popup.

Note :-

- Without handling javascript popup we will get **“UnHandledAlertException”**
- If the popup is alert then there will be no difference between accept() and dismiss() both will click on the OK button only.
- If we try to switch driver control to alert popup before it appears we will get **“NoAlertPresentException”**

2. Hidden division popup/modals/overlays

Characteristics :-

- We cannot move this popup.
- We can inspect this popup.

To handle :-

We can make use of findElement().

3. File Upload popup

Characteristics :-

- We can move this popup.
- We cannot inspect this popup.
- This popup will have open and cancel button.
- We can either avoid or handle file upload popup.

To avoid :-

Identify input tag and make use of sendKeys(“ absolute path of file ”).

To handle :-

- By using Robot class.
- By using 3rd party library called AutoIT.

4. File Download popup

By default it will be avoided by the selenium.

5. Notification popup

<u>**Characteristics :-**</u>

- We cannot move this popup.
- We cannot inspect this popup.
- This popup has an allow and block button.
- It will be displayed below the address bar in the beginning.

To avoid :-

- In order to avoid this popup we can change the settings of the bowser, so that notification popup will not be displayed.
- To change the settings of the browser we use **addArguments()** of **ChromeOptions** class.

Syntax :- ChromeOptions setting=new ChromeOptions();

setting.addArguments("--disable-notifications");

or

setting.addArguments("--incognito");

Note :- List of Chromium commands is available in google.

To handle :-

By using Robot class.

6. Authentication Popup

Characteristics :-

- We cannot move this popup.
- We cannot inspect this popup.
- This popup will have a username and password text box along with sign-in and cancel button.

To avoid :-

We can handle this popup by sending username and password along with the URL inside the get().

7. Child Window popup

Characteristics :-

- We can move the popup.
- We can inspect the popup.
- This popup has minimize , maximize and close the button along with the address bar.

To Handle :-

We can make use of getWindowHandle(), getWindowHandles() and driver.switchTo().window()

JavascriptExecutor

- JavascriptExecutor is an interface, which provides the mechanism to write the JavaScript code in selenium webdriver
- In case, findElements and locators is not working we can use JavascriptExecutor to perform an action on a web element.
- It provides 2 abstract methods :
    1. executeScript()
    2. executeAsyncScript()
- These two methods are implemented in RemoteWebDriver class
- To use these methods we should cast the WebDriver object to JavascriptExecutor type.
- Syntax :- JavascriptExecutor js = (JavascriptExecutor) driver ;

  js.executeScript() ;

Note :- executeScript() receives two inputs

1. JavaScript code in String format.
2. Generic type which acts as input to JavaScript code.

In JavaScript code we can use arguments variable to access input.

Operations performed by using JavascriptExecutor :-

1. Window scrolling
2. Performing action on disabled element.
3. Performing action on hidden element.

In JavaScript by default two global objects will be available.

1. Window object represented by reference variable window.
2. Document object represented by reference variable document.

Using these two global objects we can invoke functions in JavaScript.

To scroll there are 3 ways:-

scrollBy(int x , int y) :- this method is used to scroll the window horizontally or vertically based on x and y co-ordinates.

scrollTo(int x , int y) :- this method is used to scroll the window to a particular co-ordinate from any position.

scrollIntoView(Boolean) :- this method is used to move the element to the visible area of the page.

Note :- To scroll the window to extreme bottom of the webpage

```
window.scrollTo( 0, document.body.scrollHeight) ;
```

Data Driven Testing

- Testing the application with multiple data which is kept in external resource files like Excel, Property file etc. is called data driven testing.

Advantages of data driven testing

1. Maintenance of test data in Excel or property file is easier.
2. Modification of test data is easier.
3. Reusability of test data
4. Test data can be created before the test execution.
5. We can test the application with huge volume of data.

Property File

- Property file is used to store the common data. In order to create property file we can make use of notepad and save it as any file name with .properties extension
- In property file , data will be stored in form of key value pair.
- Key and value should be separated by :
- By default all the data available in property file is a String.

Advantages of property file :-

- It is very fast in execution.
- It is very light weight compare to any other external resource file.

Steps to fetch data from property file

Step 1 :- Create an object of FileInputStream / FileReader

```
        FileInputStream fis = new FileInputStream(“File path”);
```

Step 2 :- Create File type Object

```
        Properties prop = new Properties();
```

Step 3 :- Call read methods

```
        prop.load( fis );
        prop.get(“ Key ”);
```

Excel file

- Whenever we want to read the data from Excel we need to use Apache POI plugin so that we can read the data from all Microsoft documents like .xls and .xlsx
- Apache POI is a free and open source library tool similar to selenium

Steps to install Apache POI

1. Go to url “ https://poi.apache.org/download.html ”
2. Click on Binary Artifacts
3. Click on poi-bin-5.2.3-zip
4. Extract the zip file.
5. Copy the jars present in lib , ooxml-lib folder and in poi-bin folder and paste it in jars folder (21 jars)
6. Add all these jars to build path.

Steps to fetch data from Excel file

Step 1 :- Create an object of FileInputStream / FileReader

```
FileInputStream fis = new FileInputStream(“File path”);
```

Step 2 :- Create File type Object

```
Workbook workbook = WorkbookFactory.create( fis );
```

Step 3 :- Call read methods

To fetch data in String format :-

```
workbook.getSheet(“sheet name”).getRow(index).getCell(index).toString();
```

To fetch data in numerical format :-

```
workbook.getSheet(“sheet name”).getRow(index).getCell(index).getNumericCellValue();
```

To fetch data in date format :-

```
workbook.getSheet(“sheet name”).getRow(index).getCell(index).getLocalDateTimeCellValue();
```

To fetch data in Boolean format :-

```
workbook.getSheet(“sheet name”).getRow(index).getCell(index).getBooleanCellValue();
```

Page Object Model (POM)

- POM is one of the java design pattern which is used to store the elements.
- We write separate java class for each and every page of the web application.
- It is also called as element repository of a particular web page.
- It is also used to avoid **StaleElementReferenceException**.
- In POM we declare the element by using **@FindBy** annotation and we initialize the element by using PageFactory.initElements() method.

Note :- PageFactory.initElements() method will take 2 arguments

1. WebDriver (driver)
2. Object of POM class or page class (this)

To find element

Syntax :- @FindBy (locator name = “ locator value ”)
private WebElement elementName;

Note :- Provide getter method.

Advantages of POM

1. Maintenance of web elements is easy, since web elements are maintained based on the page.
2. Modification of web element is easy whenever the UI is changed.
3. We can avoid StaleElementReferenceException.
4. Object repositories can be easily shared across the team members via GitHub.

Note :- In real time the number of POM classes will be equal to the number of web pages present in the application.