Created By - VIVEK KUMAR (03/01-2026)

Selenium Locators

BASIC LOCATORS/PREDEFINED

For locating Single Web Element - (.findElement)

---

1) - **id** -- `driver.findElement(By.id("email")).sendKeys("asjerrt");`

2) - **name** -- `driver.findElement(By.name("pass")).sendKeys("544fy");`

3) - **Link Text** -- `driver.findElement(By.linkText("Create a page")).click();`

4) - **Partial Link Text** `--driver.findElements(By.partialLinkText("Createa")).click();`

For Locating Multiple Web Elements - (.findElements)

---

5) - **Tag Name** --

```
List<Webelement> pho = driver.findElements(By.tagName("img"));
            System.out.println("Total photos on Page : " +pho.size());
```

6) - **Class Name** --

```
List<WebElement> linku=driver.findElements(By.className("trytr"));
             System.out.println("Total Link :"linku.size());
```

CUSTOMIZED Locators

7) - **CSS Selectors**

8) - **XPath**

****

CSS Selectors

****

tag id -- tag#id

tag class -- tag.class

tag attributes -- tag[attribute='value']

tag class attributes -- tag.class[attribute='value']

****

XPath (XML Path Language - Query Language, used to find elements in HTML or XML)

****

Absolute XPath(Full XPath) -- Starts with single slash / -- not much stable

```
/html/body/div[1]/div/div[1]/div/div[1]/header/div[1]/div[1]/a/picture/img
```

Relative XPath(Partial XPath) -- Starts with Double slash // -- much stable

.....

XPath using Attributes

XPath using Single Attribute

```
-- //tag[@attributes='value']
```

XPath using multiple attributes (and)

```
-- //tag[@attributes1='value' and @attributes2='value']
```

XPath using multiples attributes (or)

```
-- //tag[@attributes1='value' or @attributes2='value']
```

XPath using functions

Xpath using contains() -- (*value kahin bhi ho)*

```
-- //tagname[contains(@attribute, 'substring')]
```

Xpath using starts-with() -- (*value starting me ho)*

```
-- //tagname[starts-with(@attribute, 'prefix')]
```

Xpath using text()

```
-- //tagname[text()='exact text']
-- //tagname[normalize-space()='text']
```

Xpath using contains() and text()

```
-- //tagname[contains(@attribute, 'substring') and text()='exact text']
```

Xpath using starts-with() and text()

```
-- //tagname[starts-with(@attribute, 'prefix') and text()='exact text']
```

XPath using Index - (in Selenium, Index starts with 1)

Xpath Using Index

```
-- (//tag[@attribute="value"])[i]      or     (//tag)[i]
```

Chained XPath - Combination of Relative XPath + Absolute XPath

Chained Xpath

```
-- //tag[@attribute="value"]/div/div/div[2]/input
```

XPath Axis -

self Selects the current node itself

```
-- //tag[@attribute="value"]/self::input
```

parent Selects the parent of the current node

```
-- //tag[@attribute="value"]/parent::input
```

child Selects all children of the current node

```
-- //tag[@attribute="value"]/child::div
```

ancestor Selects all ancestors (parent, grandparent, etc.)

```
-- //tag[@attribute="value"]/ancestor::div
```

ancestor-or-self Selects all ancestors + self node

```
-- //tag[@attribute="value"]/ancestor-or-self::div
```

descendant Selects all descendants (children, grandchildren)

```
-- //tag[@attribute="value"]/descendant::div
```

Descendant-or-self Selects all descendants + self node

```
-- //tag[@attribute="value"]/descendant-or-self::div
```

preceding-sibling Selects all siblings before the current node

```
-- //tag[@attribute="value"]/preceding-sibling::div
```

following-sibling Selects all siblings after the current node

```
-- //tag[@attribute="value"]/following-sibling::div
```

following selects all nodes after the current node

```
-- //tag[@attribute="value"]/following::div
```

Preceding Select all nodes before the self node(-ancestors)

```
-- //tag[@attribute="value"]/preceding::div
```

XPath Axis Refinement -

Wild Card (*)

```
-- //tag[@attribute="value"]/child::div
```

using Wild card `-- //tag[@attribute="value"]/child::*`

Short form (only for descendants)

```
//tag[@attribute="value"]/descendant::div --
//tag[@attribute="value"]//div
```

Index Function **-**

(Suppose total index is - 6)

`[last()]`

```
(//tag[@attribute="value"]/descendant::input)[6]
(//tag[@attribute="value"]/descendant::input)[last()]      // returns index 6
```

`[last()-1]`

```
(//tag[@attribute="value"]/descendant::input)[last()-1]// returns index 5
```

`[position()]`

```
(//tag[@attribute="value"]/descendant::input)[1]
(//tag[@attribute="value"]/descendant::input)[position()=1] // returns index 1
```

Even Elements `[position() mod 2 = 0]`

Only selects elements at even positions - (2nd, 4th, 6th...)

```
(//tag[@attribute="value"]/descendant::input)[position() mod 2 = 0]
```

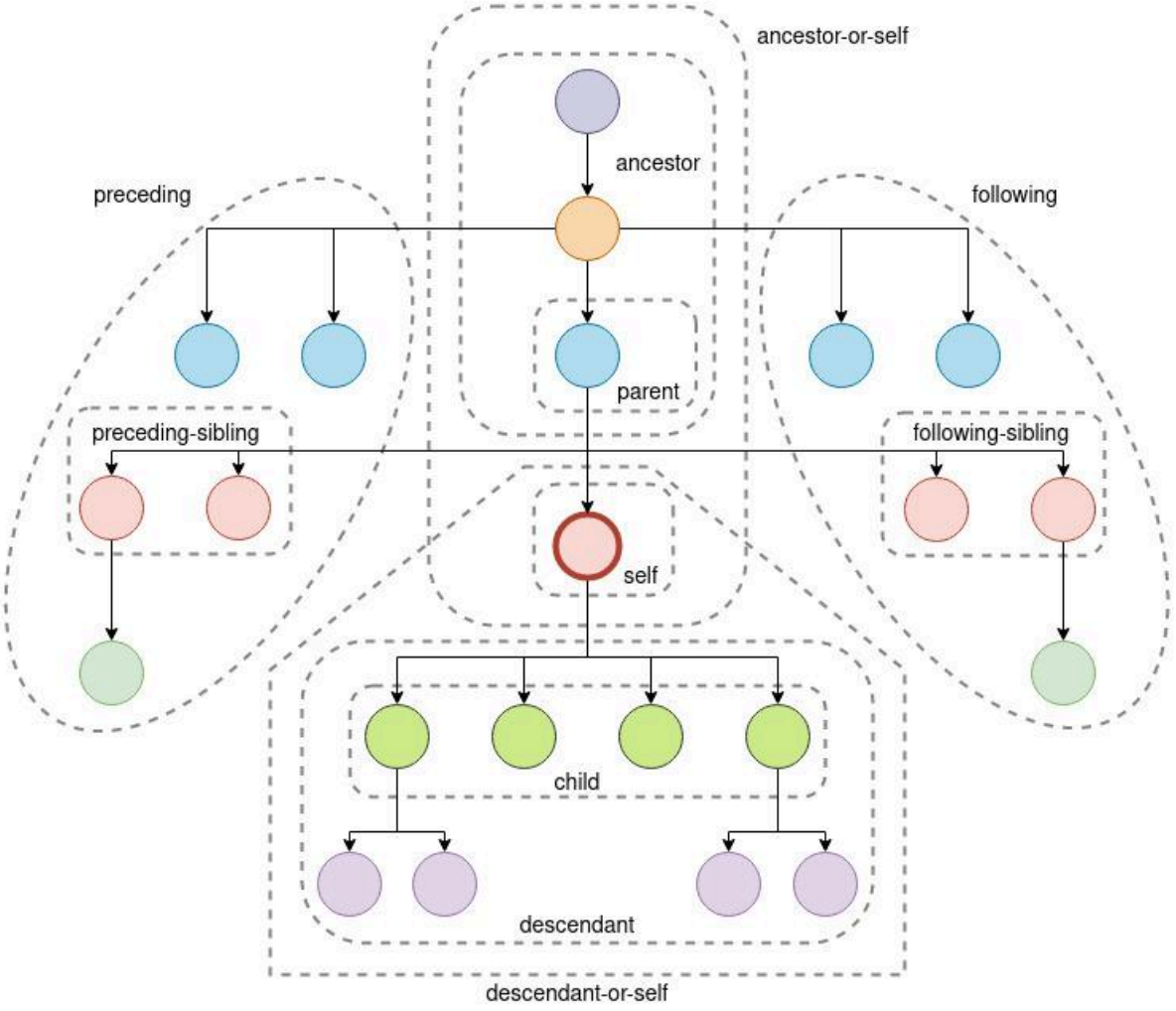
Use case: `Zebra tables, alternate rows, repeated UI patterns`

Odd Elements `[position() mod 2 = 1]`

Only selects elements at odd positions - (1st, 3rd, 5th...)

```
(//tag[@attribute="value"]/descendant::input)[position() mod 2 = 1]
```

Use case: `Zebra tables, alternate rows, repeated UI patterns`



Relative Locators

- **Relative Locators**: Selenium 4 में यह नया फीचर जोड़ा गया है।
- काम करने का तरीका: आप किसी element को उसके आसपास मौजूद दूसरे elements के आधार पर locate कर सकते हैं (जैसे *above, below, near, toLeftOf, toRightOf*).
- फायदा: जब किसी element का unique locator उपलब्ध न हो, तब relative locators से उसे आसानी से identify किया जा सकता है।

🗂️ Relative Locators

Locator	अर्थ **(Meaning)**	उदाहरण **(Example in Hindi)**

above()	किसी element के ऊपर मौजूद element को ढूँढना	"Password textbox के ऊपर जो label है उसे locate करो"
below()	किसी element के नीचे मौजूद element को ढूँढना	"Username textbox के नीचे जो input field है उसे locate करो"
toLeftOf()	किसी element के बाएँ तरफ मौजूद element को ढूँढना	"Login button के left side में जो checkbox है उसे locate करो"
toRightOf()	किसी element के दाएँ तरफ मौजूद element को ढूँढना	"Email textbox के right side में जो icon है उसे locate करो"
near()	किसी element के पास (करीब) मौजूद element को ढूँढना	"Search box के पास जो magnifying glass icon है उसे locate करो"

✨ सारांश

Relative Locators का फायदा यह है कि जब किसी element का unique locator (जैसे ID या XPath) उपलब्ध नहीं होता, तब आप उसके आसपास के elements के आधार पर उसे आसानी से पहचान सकते हैं।

```
above()
below()
toLeftOf()
toRightOf()
near()
```

```
driver.findElement(RelativeLocator.with(By.tagName("button")).near(mytext)
).click();
```

Created By - Vivek Kumar

Handling with webelements

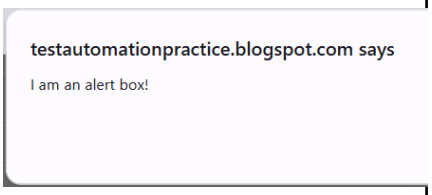
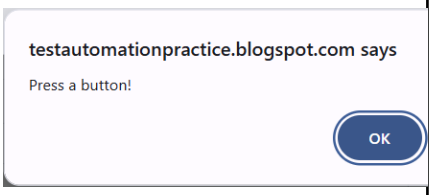
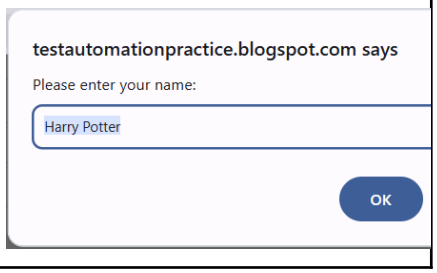
Checkbox

Sunday Monday Tuesday Wednesday Thursday
Friday Saturday

- Just Find Element and use .Click()
- Use Loop for selecting all options.

Alerts

[Browser Alerts (JavaScript Alerts)]
Alerts - alert windows are not web elements.

Normal/simple alert	confirmation alert	prompt alert
		

Normal/simple alert -- with Ok Button --
`driver.switchTo().alert().accept();`
confirmation alert -- with OK & Cancel Button --
`driver.switchTo().alert().dismiss();`
prompt alert -- input box (only one) --

`mywt.until(ExpectedConditions.alertIsPresent()).accept();`

Authenticated alert -- username & password -- URL injection Process
`https://username:password@the-internet.herokuapp.com/basic_auth`

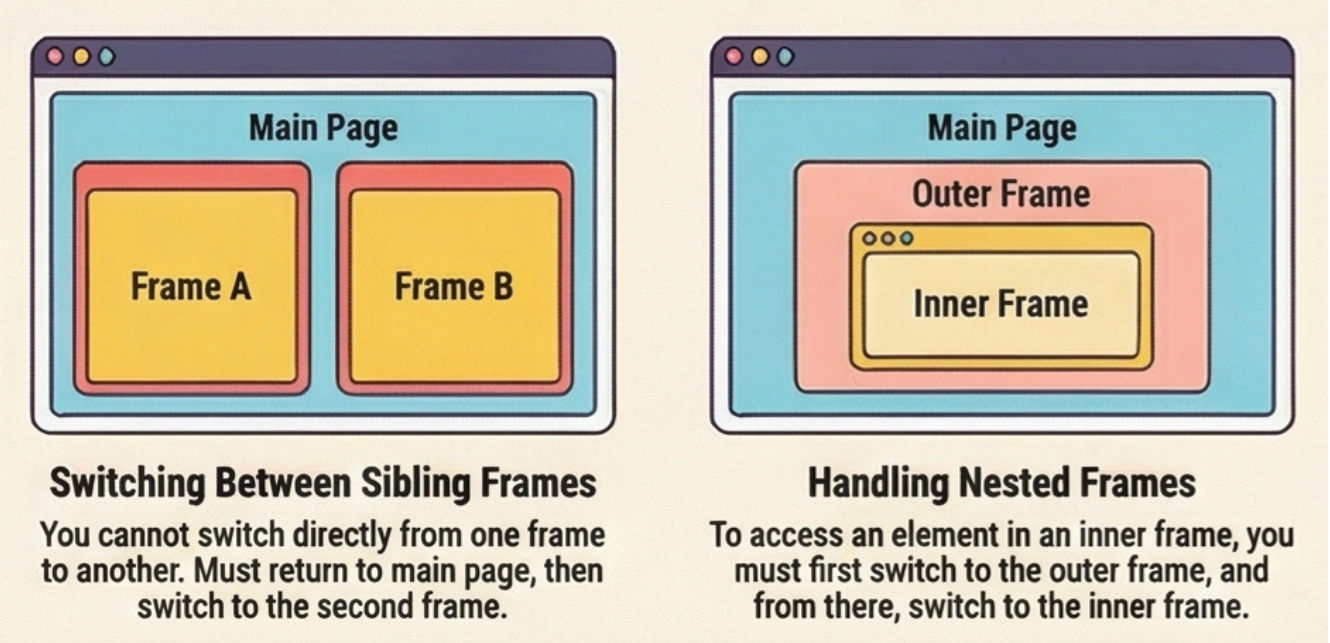
Username - admin
Password - admin

```
https://the-internet.herokuapp.com/basic_auth                    -- base
URL
https://admin:admin@the-internet.herokuapp.com/basic_auth        -- URL
injection
```

Frame/iFrame (inner Frame)

Use This Tags - <frame>, <iframe> sometimes <form>

```
driver.switchTo().frame(name)
driver.switchTo().frame(id)
driver.switchTo().frame(WebElement)
driver.switchTo().frame(index)
```



`driver.switchTo().defaultContent()`	`driver.switchTo().parentFrame()`

ElementNotInteractableException

👉 Selenium bol raha hota hai:

> **“Element DOM me hai, par user jaise interact karna possible nahi hai.”**

Common reasons:

- Element **hidden** hai (`display:none`, `visibility:hidden`)
- Element **overlay / ads / loader** ke peeche hai
- Element **off-screen** hai
- Page abhi **ready state** me nahi hai

** selenium click() is like human click, but JavascriptExecutor click() is DOM event.*

Use JavascriptExecutor class

```
JavascriptExecutor    js= (JavascriptExecutor) driver;  //downcasting
js.executeScript("arguments[0].click();", link);    // use executeScript ()
```

Dropdown Box

Select dropdown
non-select/bootstrap dropdown
Hidden dropdown
Auto-suggest dropdown

Select dropdown

*Find out by <select> tag

Use Select Class.

```
Select sc = new Select(     );
```

//Step 1 - Find dropdown element and capture in variable.
//Step 2- create object of select class and pass variable as parameter.
//Step 3- use methods to select desired options.

Methods
selectByVisibalText()
selectByValue()

selectByIndex()
getOptions() - returns all the options as WebElement

```
public static void main(String[] args) {

            WebDriver driver=new EdgeDriver();        //browser launch
            driver.manage().window().maximize();      //maximize browser
            driver.manage().deleteAllCookies();       //deleta all cookies
            driver.get("https://testautomationpractice.blogspot.com/");

      //Step 1 - Find dropdown element and capture in variable(cou).
      WebElement cou=driver.findElement(By.xpath("//select[@id=\"country\"]"));

      //Step 2- create object of select class.
            Select sc=new Select(cou);

            //Step 3- use methods to select desired options.
            //sc.selectByVisibleText("India");
            //sc.selectByValue("uk");
            sc.selectByIndex(3);

            //for all options
            List<WebElement> allop=sc.getOptions();

            //validation for Total No. of options
            int cv=allop.size();           //current value
            int ev=10;                    //expected value
            if(cv==ev) {System.out.println("Total No. of options : " +
allop.size()+ " - Test Pass");}
            else {System.out.println("fail");}

            //Print All Options.
            for(int i=0;i<allop.size();i++) {
                  String countr=allop.get(i).getText();
                  System.out.println(countr);
            }}
```

Mouse Actions

- Use Actions class (Not Action, Actions
- `Actions ac= new Actions(driver);`

Mouse hover -- `.moveToElement(WebElement).performe()`
right click/context click -- `.contextClick(WebElement).performe()`
double click -- `.doubleClick(WebElement).performe()`

drag and drop -- `.dragAndDrop(source, target).performe()`
Slider

Slider

Step1. Capture the element.
Step2. Find position of element -- `.getLocation()`
Step3. drag the element -- `.dragAndDropBy(WebElement, x,y)`

Keyboard Actions

```
ac.keyDown(Keys.CONTROL).sendKeys("a").keyUp(Keys.CONTROL).perform();
```

JavascriptExecutor (Interface)

```
JavascriptExecutor  js= (JavascriptExecutor)driver; // Downcasting
```

.click()

```
-- js.executeScript("arguments[0].click()",webelement);
```

.sendKeys()

```
-- js.executeScript("arguments[0].setAttribute('value', 'nitin')",webelement);
```

Scrolling Page

it's part of browser not web app.

Scroll to Pixel

```
-- js.executeScript("window.scrollBy(x,y)");
```

Scroll till the bottom

```
-- js.executeScript("window.scrollBy(0,document.body.scrollHeight)");
```

Scroll to element

```
-- js.executeScript("arguments[0].scrollIntoView()",webelement);
```

File Uploading

*** Find Click Button**
*** Use .sendKeys();**

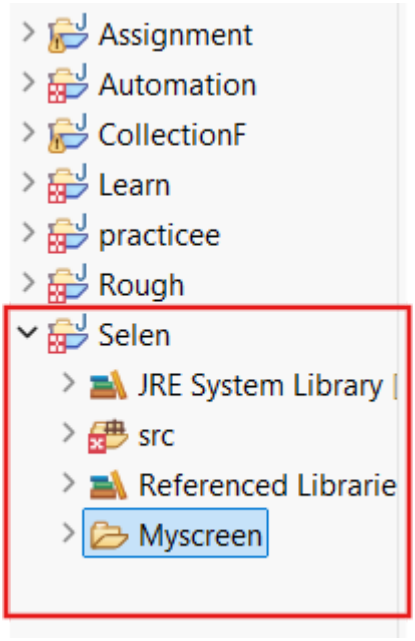
```
WebElement butto=driver.findElement(By.id("singleFileInput"));

butto.sendKeys("C:\\Users\\420st\\OneDrive\\Desktop\\sel\\qaTest.txt");
```

Capture ScreenShot

- Use TakesScreenshot (interface)

`TakesScreenshot ts= (TakesScreenshot)driver;` `//downcasting`

1) Full page Screenshot
2) Specific area of the page
3) web element

Create a folder(Myscreen) in your Project	

Full page Screenshot

```
//capture screenshot and save into variable.
```

```
File capturefile=ts.getScreenshotAs(OutputType.FILE);

//give location of folder
String path = System.getProperty("user.dir");
//project ka path
File dest = new File(path + "\\Myscreen\\ramfullscreen.png");
//folder + file

//copy capturefile to target folder
capturefile.renameTo(dest);
```

Specific area of the page

```
 //capture Element of area.
WebElement
butto=driver.findElement(By.id("singleFileInput"));

//capture screenshot and save into variable.
File capturefile=butto.getScreenshotAs(OutputType.FILE);

//give location of folder
String path = System.getProperty("user.dir");
//project ka path
File dest = new File(path + "\\Myscreen\\ramfullscreen.png");
//folder + file

//copy capturefile to target folder
capturefile.renameTo(dest);
```

Options

ChromeOptions
EdgeOptions
FirefoxOptions

These classes are used to customise browser behaviour* ***before launching the browser.*

1) Headless Testing

```
EdgeOptions ed=new EdgeOptions();
ed.addArguments("--headless=new");              //headless

WebDriver driver=new EdgeDriver(ed);
```

2) incognito mode

```
ChromeOptions ed=new ChromeOptions();
```

```
ed.addArguments("--incognito");            //incognito

WebDriver driver=new ChromeDriver(ed);
```

3) SSL certificate

```
ChromeOptions ed=new ChromeOptions();
ed.setAcceptInsecureCerts(true);                //SSL

WebDriver driver=new ChromeDriver(ed);
```

4) Remove "Chrome is being controlled by....."

```
ChromeOptions ed=new ChromeOptions();
ed.setExperimentalOption("excludeSwitches", new String[]
{"enable-automation"});

WebDriver driver=new ChromeDriver(ed);
```

5) Enable extension (.crx)

```
ChromeOptions ed=new ChromeOptions();
File crxfile=new
File("C:\\Users\\420st\\OneDrive\\Desktop\\sel\\AdBlocker-Ultimate.crx");
ed.addExtensions(crxfile);
```