

| **Web Driver Commands** | **Syntax and Description** |
| --- | --- |
| 1. get(String url) command | **Syntax:**  driver.get(URL);  It loads a new web page in the current browser window and accepts a string parameter that specifies the URL of the web page to be loaded. |
| 2. getTitle() Command | **Syntax:**  driver.getTitle();  It gets the title of the current web page displayed in the browser. It does not accept any parameters. It returns the title of the specified URL as a string. |
| 3. getCurrentUrl() Command | **Syntax:**  driver.getCurrentUrl();  It gets the URL of the current web page shown in the browser. It does not accept any parameters and returns the URL as a string. |
| 4. getPageSource() Command | **Syntax:**  driver.getPageSource();  It gets the entire page source of the current web page loaded in the browser. It does not accept any parameters, but it does return the page source as a string. |
| 5. close() Command | **Syntax:**  driver.close();  It closes the current browser window or tab. Also, this command does not accept any types of parameters. It also does not return anything. |
| 6. quit() Command | **Syntax:**  driver.quit();  It closes all the browser windows and tabs for a particular WebDriver session. This command does not accept any parameters and not return anything. |
| 7. to() Command | **Syntax:**  driver.navigate().to(URL);  Loads a new web page in the current browser window. It accepts a string parameter that specifies the URL of the web page to be loaded. |
| 8. back() Command | **Syntax:**  driver.navigate().back();  Moves back one step in the browser’s history stack. It does not accept any parameters and does not return anything. |
| 9. forward() Command | **Syntax:**  driver.navigate().forward();  Moves forward one step in the browser’s history stack. It does not accept any parameters and does not return anything. |
| 10. refresh() Command | **Syntax:**  driver.navigate().refresh();  Reloads the current web page in the browser window. It does not accept any parameters and does not return anything. |
| 11. sendKeys() commands | **Syntax:**  // Create WebElement WebElement temp = driver.findElement(By.id("TextBox")); // Perform sendKeys operation temp.sendKeys("GeeksForGeeks");  Entre text automatically into editable field while executing tests. these field are identified using locators like element id, name, class name, etc. |
| 12. isDisplayed() command | **Syntax:**  WebElement element = driver.findElement(By.id("gfg")); boolean status = element.isDisplayed();  It verifies whether a web element is present and visible on the web page. Returns true if the element is displayed and false if not. |
| 13. isSelected() command | **Syntax:**  WebElement element = driver.findElement(By.id("software-testing")); boolean status = element.isSelected();  It used on radio, checkboxes, dropdowns to check whether element is selected or not. If the specified element is selected, the value returned is true. If not, the value returned is false. |
| 14. submit() command | **Syntax:**  WebElement element = driver.findElement(By.id("SubmitButton")); element.submit();  It used to submit forms on browser. It doesn’t require a parameter and returns nothing. |
| 15. isEnabled() command | **Syntax:**  // Create WebElement WebElement element = driver.findElement(By.id("GFG")); // Perform isEnabled operation element.isEnabled();  Used to checks if an element is enabled for interaction on the web page or not. It returns true if the element is enabled and false if not. |
| 16. getLocation() command | Syntax:  WebElement element = driver.findElement(By.id("Selenium")); Point point = element.getLocation(); //Return  System.out.println("X cordinate : " + point.x + "Y cordinate: " + point.y);  Retrieves the location of a specific web element on the page in terms of its X and Y coordinates. |
| 17. clear( ) command | **Syntax:**  WebElement ele = driver.findElement(By.id("edittext"));  // Perform clear operation ele.clear();  Used to Clears the content of text entry fields or text areas. It doesn’t require a parameter and returns nothing. |
| 18. getSize() command | Syntax:  WebElement element = driver.findElement(By.id("SubmitButton")); Dimension dimensions = element.getSize(); System.out.println(“Height :” + dimensions.height + ”Width : "+ dimensions.width);  Used to retrieves the height and width of a rendered element on the web page. |
| 19. getAttribute() command | Syntax:  WebElement element = driver.findElement(By.id("Submit")); String attValue = element.getAttribute("id"); //This will return "Submit"  Used to retrieves the value of a specified attribute of a web element. |
| 20. click() command | Syntax:  // Create WebElement WebElement ele = driver.findElement(By.id("GeeksForGeeks")); // Perform click operation ele.click();  Used to perform click operation on a web element such as a button, link, or checkbox. |

**CSS selectors**

Types of CSS Selectors in Selenium (with Examples)

There are five types of CSS Selectors in Selenium tests:

* ID
* Class
* Attribute
* Combining Attributes
* Sub String

**1.ID**

\*\*\*\*

In CSS, we can use “#” notation to select the “id” attribute of an element.

Different syntaxes for CSS with Id are as follows:

**Syntax:**

driver.findElement(By.cssSelector(“<tagname>#<id value>”));

driver.findElement(By.cssSelector(“#<id value>”));

driver.findElement(By.cssSelector(“<tagname>[id=’<id value>’]”));

**Example :**

driver.findElement(By.cssSelector("a#login2"));

driver.findElement(By.cssSelector("#login2"));

driver.findElement(By.cssSelector("a[id='login2']"));

**2. Class**

\*\*\*\*\*\*\*\*\*

In CSS, we can use “.” notation to select the “class” attribute of an element.

**Syntax:**

driver.findElement(By.cssSelector(“<tagname>.<class value>”));

driver.findElement(By.cssSelector(“.<class value>”));

driver.findElement(By.cssSelector(“<tagname>[class=’<class value>’]”));

**3.Attribute**

\*\*\*\*\*\*\*\*\*\*\*\*\*

Apart from “id” and “class”, other attributes can also be used to locate web elements using CSS selector.

**Syntax:**

driver.findElement(By.cssSelector(“<tagname>[href=’<href value>’]”));

**4. Combining Attributes**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1)id and attribute**

Syntax:

driver.findElement(By.cssSelector(“<tagname>#<id value>[href=’<href value>’]”));

**2)class and attribute**

Syntax:

driver.findElement(By.cssSelector(“<tagname>.<class value>[href=’<href value>’]”));

**5. Sub String**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1)Matching a prefix (Starts with: ^):** Locate the web element using the substring that starts with a certain value.

**Syntax**

driver.findElement(By.cssSelector(“<tagname>[<attribute>^=’prefix of the string’]”));

**Example:**

a[class^='Navbar\_logo\_']

**2)Matching a suffix (Ends with: $):** Locate the web element using the substring that ends with a certain value.

**Syntax:**

driver.findElement(By.cssSelector(“<tagname>[<attribute>$=’suffix of the string’]”));

**Example**

a[class$='26S5Y']

**xpath**

\*\*\*\*\*\*\*\*\*

Types of XPath in Selenium

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here is a quick overview of the two types of Selenium XPath:

**Absolute XPath:** Begins from the root of the HTML document and specifies the complete path to the element. It’s not as flexible and can break if the page structure changes.

**Relative XPath:** Starts from a specific element and navigates through the DOM hierarchy to locate the desired element. It’s more flexible and resilient to changes in the page structure.

XPath locators in Selenium WebDriver are used to identify elements on a web page. These locators allow complex and flexible navigation of the web page's Document Object Model (DOM). There are several types of XPath locators, each useful in different situations.

● **XPath locator by ID:** This locator allows you to identify an element by its id attribute.

**Example:**

driver.findElement(By.xpath("//\*[@id='username']"));

Note: XPath is a wildcard in the snippets helping to select unknown XML nodes

●  **XPath locator by class name:** This locator can identify elements based on their class attribute.

**Example:**

driver.findElement(By.xpath("//\*[@class='login-button']"));

● **XPath locator by name:** This locator identifies elements by their name attribute.

**Example:**

driver.findElement(By.xpath("//\*[@name='password']"));

● **XPath locator by tag name:** This locator can identify elements by their HTML tag name.

**Example:**

driver.findElement(By.xpath("//p"));

● **XPath locator by text:** This locator identifies elements based on their inner text.

**Example:**

driver.findElement(By.xpath("//\*[text()='Submit']"));

● **XPath locator using contains:** This locator can identify elements based on a substring of one of their attribute values.

**Example:**

driver.findElement(By.xpath("//\*[contains(@href,'google.com')]"));

● **XPath locator using starts-with:** This locator identifies elements whose attribute values start with a particular string.

**Example:**

driver.findElement(By.xpath("//\*[starts-with(@id,'user')]"));

● **XPath locator using ends-with:** This locator can identify elements whose attribute values end with a particular string.

**Example:**

driver.findElement(By.xpath("//\*[ends-with(@id,'name')]"));