1. CSS selector syntax by taking partial words in a parameter value.

Ex : If element in HTML is like 🡪 **<a class="" href="https://www.joecolantonio.com/" id="logo</a>**

Then **CSSSelector** to find this element is

**a[href\*=’joecola’] 🡪** here if we put \* then it will search for all the elements whose tag should be ‘a’ and attribute should have href and it’s value should **CONTAIN** ‘joecola’ in it.

If we write like **a[href=’joecola’] 🡪** then it will search for all the elements whose tag should be ‘a’ and attribute should have href and it’s value should be **EQUAL** to ‘joecola’.

| **Syntax** | **Example** | **Description** |
| --- | --- | --- |
| ^= | input[id^='ctrl'] | **Starting with:**  For example, if the ID of an element is ctrl\_12, this will locate and return elements with ctrl at the beginning of the ID. |
| $= | input[id$='\_userName'] | **Ending with:**  For example, if the ID for an element is a\_1\_userName, this will locate and return elements with \_userName at the end of the ID. |
| \*= | Input[id\*='userName'] | **Containing:**  **For example, if the ID of an element is panel\_login\_userName\_textfield, this will use the userName part in the middle to match and locate the element.** |

1. In any web development project, it is always a good practice to assign attributes such as Name, IDs, or Class to GUI elements. This makes the application more testable and conforms to accessibility standards. However, following these practices is not always possible. For such scenarios, we have to use advanced locator strategies such as CSS selector and XPath.***While CSS selector and XPath are popular among Selenium users, CSS selector is highly recommended over XPath due to its simplicity, speed, and performance***.

The following are the list of object identifier or locators supported by selenium.

* id
* Name
* Linktext
* Partial Linktext
* Tag Name
* class name
* Css
* xpath

In this article we will be focusing more on CSS and XPath.

**Locating Element Using CSS Selectors**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language such as HTML or XML.

Let's explore some basic CSS selectors that can be used in Selenium WebDriver. Selenium WebDriver's By class provides the **cssSelector()** method for locating elements using CSS selectors.

**Finding elements with absolute path**

CSS absolute paths refer to the very specific location of the element considering its complete hierarchy in the DOM. Here is an example where the Username Input field is located using the

absolute path. While providing absolute path, a space is given between the elements.

**driver.findElement(By.cssSelector("html body div div form input"));**

We can also use the previous selector in the following way by describing the direct parent to

child relationships ***with > separator***:

**driver.findElement(By.cssSelector("html > body > div > div > form > input"));**

***However, this strategy has limitations as it depends on the structure or hierarchy of the***

***elements on a page. If this changes, the locator will fail to find the element.***

**Finding elements with Relative path**

With relative path we can locate an element directly, irrespective of its location in the DOM.

For example, we can locate the Username Input field in the following way, assuming it is the

first <input> element in the DOM:

**driver.findElement(By.cssSelector("input"));**

**Finding elements using the Class Selector**

While finding elements using the CSS selector, we can use the Class attribute to locate an element. This can be done by specifying the type of HTML tag, and then adding a dot followed by the value of the class attribute in the following way:

**driver.findElement (By.cssSelector ("input.login"));**

There is also a shortcut where you can put a . and class attribute value and ignore the HTML tag.

**driver.findElement (By.cssSelector (".login"));**

**Finding elements using the ID Selector**

We can locate the element using the IDs assigned to elements. This can be done by specifying the type of HTML tag, then entering a hash followed by the value of the Class attribute, as shown:

**driver.findElement (By.cssSelector ("input#username"));**

There is also a shortcut where you can enter # and a class attribute value and ignore the HTML tag.

**driver.findElement (By.cssSelector ("#username"));**

**Locating Element Using XPATH Selectors**

XPath, the XML path language, is a query language for selecting nodes from an XML document. Locating elements with XPath works very well with a lot of flexibility. However, this is the least

preferable locator strategy due its slow performance.

One of the important differences between XPath and CSS is, with XPath we can search elements backward or forward in the DOM hierarchy while CSS works only in a forward direction. This means that with XPath we can locate a parent element using a child element.

**Performing Partial Match**

| **Syntax** | **Example** | **Description** |
| --- | --- | --- |
| startswith() | input[starts-with(@id,'ctrl')] | **Starting with:**  For example, if the ID of an element is ctrl\_12, this will locate and return elements with ctrl at the beginning of the ID. |
| endswith() | input[ends-with(@id,'\_userName')] | **Ending with:**  For example, if the ID of an element is a\_1\_userName, this will locate and return elements with \_userName at the end of the ID. |
| contains() | Input[contains(@id,'userName')] | **Containing:**  For example, if the ID for an element is panel\_login\_userName\_textfield, this will use the userName part in the middle to match and locate the element. |