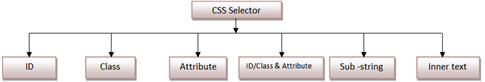
Selector Pattern is constructed using HTML tags, attributes and their values. The central theme behind the procedure to create CSS Selector and Xpath are very much similar underlying the only difference in their construction protocol.

Like Xpath, CSS selector can also locate web elements having no ID, class or Name.

So now gearing ahead, let us discuss the primitive types of CSS Selectors:

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Using-CSS-Selector-as-a-Locator.jpg)

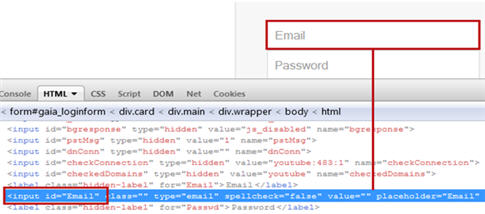
**CSS Selector: ID**

In this sample, we would access “Email” text box present in the login form at Gmail.com.

The Email textbox has an ID attribute whose value is defined as “Email”. Thus ID attribute and its value can be used to create CSS Selector to access the email textbox.

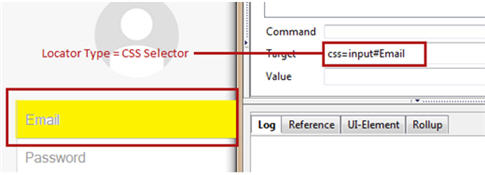
**Creating CSS Selector for web element**

**Step 1**: Locate / inspect the web element (“Email” textbox in our case) and notice that the html tag is “input” and value of ID attribute is “Email” and both of them collectively make a reference to the “Email Text box”. Hence the above data would be used to create CSS Selector.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Using-CSS-Selector-as-a-Locator-2.jpg)

**Verify the locator value**

**Step 1**: Type “css=input#Email” i.e. the locator value in the target box in the Selenium IDE and click on the Find button. Notice that the Email Text box would be highlighted.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Using-CSS-Selector-as-a-Locator-3.jpg)

**Syntax**

css=<HTML tag><#><Value of ID attribute>

* **HTML tag** – It is tag which is used to denote the web element which we want to access.
* **#**– The hash sign is used to symbolize ID attribute. It is mandatory to use hash sign if ID attribute is being used to create CSS Selector.
* **Value of ID attribute**– It is the value of an ID attribute which is being accessed.
* The value of ID is always preceded by a hash sign.

**Note:**Also applicable for other types of CSS Selectors

* While specifying CSS Selector in the target text box of Selenium IDE, always remember to prefix it with “css=”.
* The sequence of the above artifacts is inalterable.
* If two or more web elements have the same HTML tag and attribute value, the first element marked in the page source will be identified.

**CSS Selector: Class**

In this sample, we would access “Stay signed in” check box present below the login form at gmail.com.

The “Stay signed in” check box has a Class attribute whose value is defined as “remember”. Thus Class attribute and its value can be used to create CSS Selector to access the designated web element.

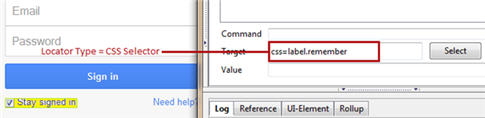
Locating an element using Class as a CSS Selector is very much similar to using ID, the lone difference lies in their syntax formation.

**Creating CSS Selector for web element**

**Step 1**: Locate / inspect the web element (“Stay signed in” check box in our case) and notice that the html tag is “label” and value of ID attribute is “remember” and both of them collectively make a reference to the “Stay signed in check box”.

**Verify the locator value**

**Step 1**: Type “css=label.remember” i.e. the locator value in the target box in the Selenium IDE and click on the Find Button. Notice that the “Stay signed in” check box would be highlighted.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Using-CSS-Selector-as-a-Locator-4.jpg)

**Syntax**

css=<HTML tag><.><Value of Class attribute>

* **.**– The dot sign is used to symbolize Class attribute. It is mandatory to use dot sign if Class attribute is being used to create CSS Selector.
* The value of Class is always preceded by a dot sign.

**CSS Selector: Attribute**

In this sample, we would access “Sign in” button present below the login form at gmail.com.

The “Sign in” button has a type attribute whose value is defined as “submit”. Thus type attribute and its value can be used to create CSS Selector to access the designated web element.

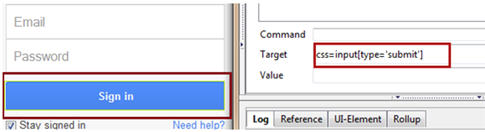
**Creating CSS Selector for web element**

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

**Step 1**: Locate / inspect the web element (“Sign in” button in our case) and notice that the html tag is “input”, attribute is type and value of type attribute is “submit” and all of them together make a reference to the “Sign in” button.

**Verify the locator value**

**Step 1**: Type “css=input[type=’submit’]” i.e. the locator value in the target box in the Selenium IDE and click on the Find Button. Notice that the “Sign in” button would be highlighted.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Using-CSS-Selector-as-a-Locator-5.jpg)

**Syntax**

css=<HTML tag><[attribute=Value of attribute]>

* **Attribute**– It is the attribute we want to use to create CSS Selector. It can value, type, name etc. It is recommended to choose an attribute whose value uniquely identifies the web element.
* **Value of attribute**– It is the value of an attribute which is being accessed.

**CSS Selector: ID/Class and attribute**

In this sample, we would access “Password” text box present in the login form at gmail.com.

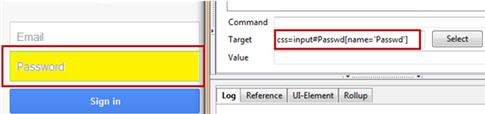
The “Password” text box has an ID attribute whose value is defined as “Passwd”, type attribute whose value is defined as “password”. Thus ID attribute, type attribute and their values can be used to create CSS Selector to access the designated web element.

**Creating CSS Selector for web element**

**Step 1**: Locate / inspect the web element (“Password” text box in our case) and notice that the html tag is “input”, attributes are ID and type and their corresponding values are ”Passwd” and “password” and all of them together make a reference to the “Password” textbox.

**Verify the locator value**

**Step 1**: Type “css=input#Passwd[name=’Passwd’]” i.e. the locator value in the target box in the Selenium IDE and click on the Find Button. Notice that the “Password” text box would be highlighted.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Using-CSS-Selector-as-a-Locator-6.jpg)

**Syntax**

css=<HTML tag><. Or #><value of Class or ID attribute><[attribute=Value of attribute]>

Two or more attributes can also be furnished in the syntax. For example, *“css=input#Passwd[type=’password’][name=’Passwd’]”.*

**CSS Selector: Sub-string**

CSS in Selenium allows matching a partial string and thus deriving a very interesting feature to create CSS Selectors using sub strings. There are three ways in which CSS Selectors can be created based on mechanism used to match the sub string.

**Types of mechanisms**

All the underneath mechanisms have symbolic significance.

* Match a prefix
* Match a suffix
* Match a sub string

Let us discuss them in detail.

**Match a prefix**

It is used to correspond to the string with the help of a matching prefix.

**Syntax**

css=<HTML tag><[attribute^=prefix of the string]>         

* **^**– Symbolic notation to match a string using prefix.
* **Prefix**– It is the string based on which match operation is performed. The likely string is expected to start with the specified string.

For Example: Let us consider “Password textbox”, so the corresponding CSS Selector would be:

*css=input#Passwd[name^=’Pass’]*

**Match a suffix**

It is used to correspond to the string with the help of a matching suffix.

**Syntax**

css=<HTML tag><[attribute$=suffix of the string]>        

* **#**– Symbolic notation to match a string using suffix.
* **Suffix**– It is the string based on which match operation is performed. The likely string is expected to ends with the specified string.

For Example: Lets again consider “Password textbox”, so the corresponding CSS Selector would be:

*css=input#Passwd[name$=’wd’]*

**Match a sub string**

It is used to correspond to the string with the help of a matching sub string.

**Syntax**

css=<HTML tag><[attribute\*=sub string]>          

* **\***– Symbolic notation to match a string using sub string.
* **Sub string** – It is the string based on which match operation is performed. The likely string is expected to have the specified string pattern.

For Example: Lets again consider “Password textbox”, so the corresponding CSS Selector would be:

*css=input#Passwd[name$=’wd’]*

**CSS Selector: Inner text**

Inner text helps us identify and create CSS Selector using a string pattern that the HTML Tag manifests on the web page.

Consider, “Need help?” hyperlink present below the login form at gmail.com.

The anchor tag representing the hyperlink has a text enclosed within. Thus this text can be used to create CSS Selector to access the designated web element.

**Syntax**

css=<HTML tag><:><contains><(text)>

* **:**– The dot sign is used to symbolize contains method
* Contains – It is the value of a Class attribute which is being accessed.
* Text – The text that is displayed anywhere on the web page irrespective of its location.

This is one of the most frequently used strategies to locate web element because of its simplified syntax.

*Owing to the fact that creating CSS Selector and Xpath requires a lot of efforts and practice, thus the process is only exercised by more sophisticated and trained users.*

[**Next Tutorial #7**](http://www.softwaretestinghelp.com/locate-elements-in-chrome-ie-selenium-tutorial-7/): Proceeding ahead with our next tutorial, we would take the opportunity to introduce you with an extension of locating strategies. Thus, in the next tutorial, we would study the**mechanism to locate web elements on Google Chrome and Internet Explorer.**

We are covering Selenium Locators in more details as it is important part of Selenium Script creation.