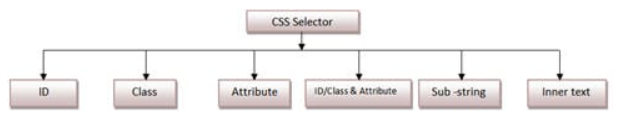
Six types of locators for identifying web elements on a web page

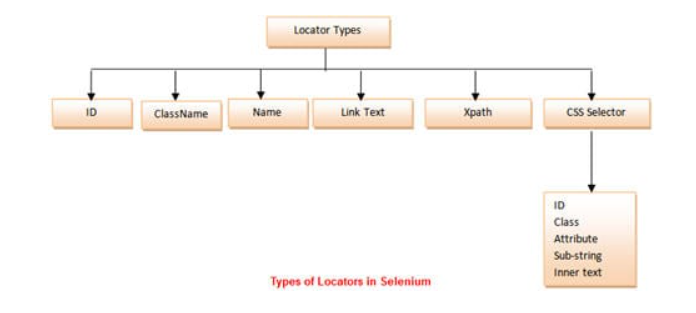
* ID
* ClassName
* Name
* Link Text
* XPath
* css selector

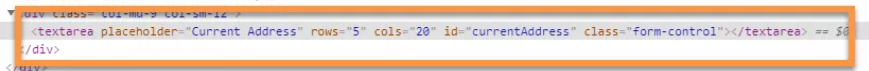
CSS Selector is the combination of an element selector and a selector value which identifies the web element within a web page. The composite of an element selector and selector value is known as **Selector Pattern**.

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.







***How to Combine the ID and other Attributes of the web element to create a CSS Selector?***

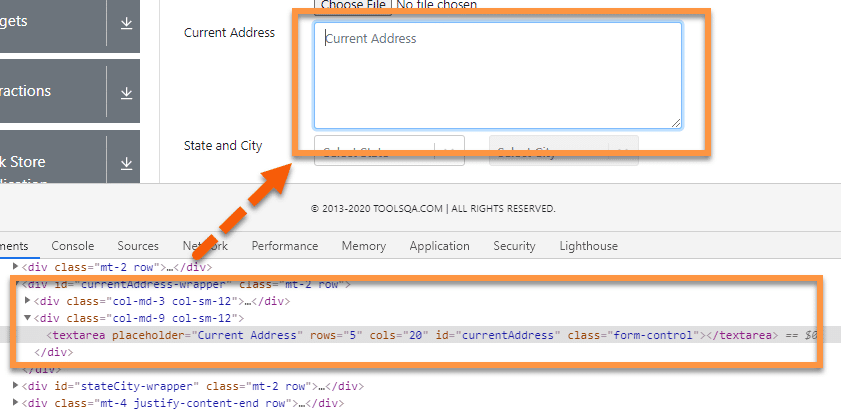
In the given element, the *HTML* structure contains a *textarea tag, id, and placeholder attribute*. We will use these together to create a *CSS Selector* statement that can easily recognize that element. So, the *CSS*

textarea**#currentAddress[placeholder='Current Address']**  
**textarea.form-control[placeholder='Current Address']**

#### How to locate a web element when one element is the **direct parent/child** of another element?

CSS Selectors allow you to select an element by using the locator of the parent element and then moving to the child element. The CSS Selector for locating the child element can be syntactically represented as follows:

|  |
| --- |
| **Parent\_locator > child\_locator** |



In the above example, we have a “**textarea**” HTML tag enclosed in the bracket, which is the child tag of “***div.”*** Assume a scenario where we are not able to identify the “**textarea**” by using its attributes, but we can identify its parent HTML tag, then we can use it to access the child tag. Let’s create the CSS selector for locating the ***textarea*** element:

|  |
| --- |
| **div>textarea[placeholder='Current Address']** |

Here we have first used the ***locator for a parent***then “**>**” followed by ***the child locator***. Similarly, this can be extended to the sub child also by adding another “**>**” followed by another locator.

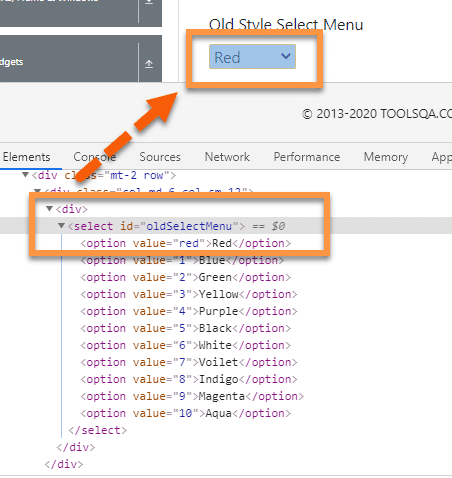
#### How to locate a web element when the element exists in the hierarchy?

Similar to the child and sub-child, we can also use a CSS Selector to select the ***nth-child of an HTML tag***. It is quite useful in recognizing list elements or in scenarios where a parent has multiple child elements with non-consistent attributes.

The syntax for locating the nth-child will be:

**Parent CSS locator >** **Child HTML tag** **: nth-of-type**(index)

Selecting nth-child using CSS Selector, for this, we will be using the following site link: [***https://www.demoqa.com/select-menu***](https://www.demoqa.com/select-menu)***.***



Let’s take the above example; we will try to find the ***CSS Selector***for the child element of “**ul**” HTML tag, i.e.    “**li”** Say, we want to find the 2nd child element of the “**ul”** then the CSS Selector expression for the same will be:

|  |  |
| --- | --- |
|  | **Parent CSS locator >** **Child HTML tag** **: nth-of-type**(index)  **select#oldSelectMenu>option:nth-of-type**(2) |

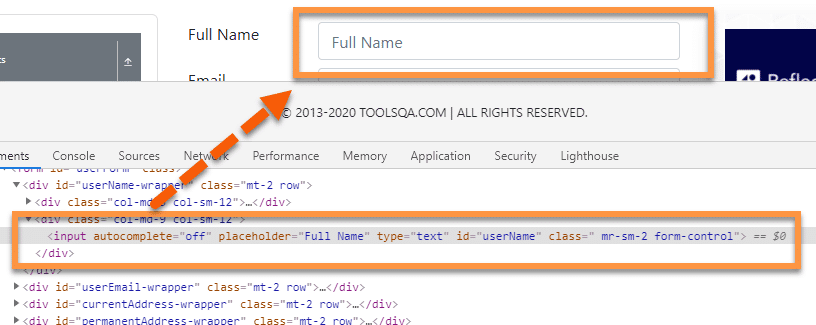
Here we started with the parent **CSS Selector** tag, followed by “**>”** which is followed by the ***HTML***  tag of the child. The child ***HTML tag is then followed by id symbol –***“***:*** ”, which is followed by “***nth-of-type(index)”*** where bracket accepts the index of the required element.

So this way, we can locate any of the HTML elements in the hierarchy.

### *****How to locate a web element using text strings?*****

Similar to XPath, ***CSS Selector*** also allows users to locate elements by using partial strings. It uses different symbols to represent the start, end, and the contents inside a text. Let’s have a look at some examples to understand more about ***CSS Sub-Strings*** in detail.

For example, we will use the following element, i.e., “**Full Name,**” as marked in the below image for locating the text box on the page.



#### How to locate a web element using the starting text?

We can locate an element by using the starting text of the element. It is quite useful if you know the starting text of the element attribute. We can use the starting character sequence of the attribute value to locate the element using CSS Selectors.

The Symbol for representing the *starting text* of a string is: **‘^’**

Using this symbol in the *CSS Selector*, the expression for locating the web element will be

|  |  |
| --- | --- |
|  | input[**id^=**'userN'] |

Here, we have used the ***id*** attribute. The value of the *id* attribute in the *HTML* is “***userName.”*** In this expression, we have used the ***first five characters*** of the expression. We can use any number of characters from the beginning.

***How to locate a web element using the* Ending text?**

Similar to the starting text, we can also use ***ending text***to recognize the element. A sequence of the ending character of the attribute value can locate any web element.

The Symbol for representing the *ending text* of a string is: ‘$’

Using this symbol in the *CSS Selector*, the expression for locating the web element will be:

|  |  |
| --- | --- |
|  | input[**id$=**'ame'] |

Here, again we have used the ***id*** attribute whose value is “***username***”. Here we have used the *last three characters* of the attribute value. The attribute and value can change as per the scenario.

***How to locate a web element using the* contains text?**

Other than *starting and ending*, the *CSS Selector* in Selenium is also available **with contains text().** It can locate the element by using *any sequential characters* from the attribute value.

The Symbol for representing the *contains the text*: **‘\*’**

Using the same symbol in *CSS Selector*, the expression for the above elements will be:

|  |  |
| --- | --- |
|  | input[**id\*=**'erNa'] |

Here, we have used the ***middle characters***of the *id* attribute value, i.e., “***username***” from the *HTML* to locate the element.

### Inner Text Conundrum:

One of the most easily readable syntaxes yet powerful. You can make use of this mechanism to identify text anywhere on the DOM using a string pattern.

Syntax:

### Inner Text Conundrum:

One of the most easily readable syntaxes yet powerful. You can make use of this mechanism to identify text anywhere on the DOM using a string pattern.

Syntax:

Inner Text Conundrum:

One of the most easily readable syntaxes yet powerful. You can make use of this mechanism to identify text anywhere on the DOM using a string pattern.

Syntax:

<HTML Tag> <**:**> <**contains**> < (text) >

div**:contains(**"^lambdatest$"**)**