Locators

A locator is a way to identify elements on a page. It is the argument passed to the [Finding element](https://www.selenium.dev/documentation/webdriver/elements/finders/) methods.

Locating elements in Selenium WebDriver is performed with the help of findElement() and findElements() methods provided by WebDriver and WebElement class.

* findElement() returns a WebElement object based on a specified search criteria or ends up throwing an exception if it does not find any element matching the search criteria.
* findElements() returns a list of WebElements matching the search criteria. If no elements are found, it returns an empty list.

The following table lists all the Java syntax for locating elements in Selenium WebDriver.

|  |  |  |
| --- | --- | --- |
| **Method** | **Syntax** | **Description** |
| By ID | driver.findElement(By.id (<element ID>)) | Locates an element using the ID attribute |
| By name | driver.findElement(By.name (<element name>)) | Locates an element using the Name attribute |
| By class name | driver.findElement(By.className (<element class>)) | Locates an element using the Class attribute |
| By tag name | driver.findElement(By.tagName (<htmltagname>)) | Locates an element using the HTML tag |
| By link text | driver.findElement(By.linkText (<linktext>)) | Locates a link using link text |
| By partial link text | driver.findElement(By.partialLinkText (<linktext>)) | Locates a link using the link's partial text |
| By CSS | driver.findElement(By.cssSelector (<css selector>)) | Locates an element using the CSS selector |
| By XPath | driver.findElement(By.xpath (<xpath>)) | Locates an element using XPath query |

By XPath

XPath stands for XML path language. It is a query language for selecting nodes from an XML document. XPath is based on the tree representation of XML documents and provides the ability to navigate around the tree by selecting nodes using a variety of criteria.

By CSS

The CSS is used as a method to identify the webobject, however NOT all browsers support CSS identification.

WebElement loginButton = driver.findElement(By.cssSelector("input.login"));

## **Relative Locators**

**Selenium 4** introduces Relative Locators (previously called as Friendly Locators). These locators are helpful when it is not easy to construct a locator for the desired element, but easy to describe spatially where the element is in relation to an element that does have an easily constructed locator.

\* [***DOM***](https://www.w3.org/TR/WD-DOM/introduction.html) (Document Object Model),

1.  ***DOM*** can be accessed in Google Chrome either by pressing ***F12*** or by ***right click*** on the web page and then by selecting ***Inspect***

By.id("firstName");

By.name("gender");

By.className("practice-form-wrapper");

By.linkText("Home");

By.tagName("a");

By.xpath("//input[@id='userName']");

By.cssSelector("input[id= ‘userName’]");

The basic syntax of identifying a web element using ***CSS*** is as follows:

css=(HTML Page)[Attribute=Value]

CSS Selector

CSS Selector

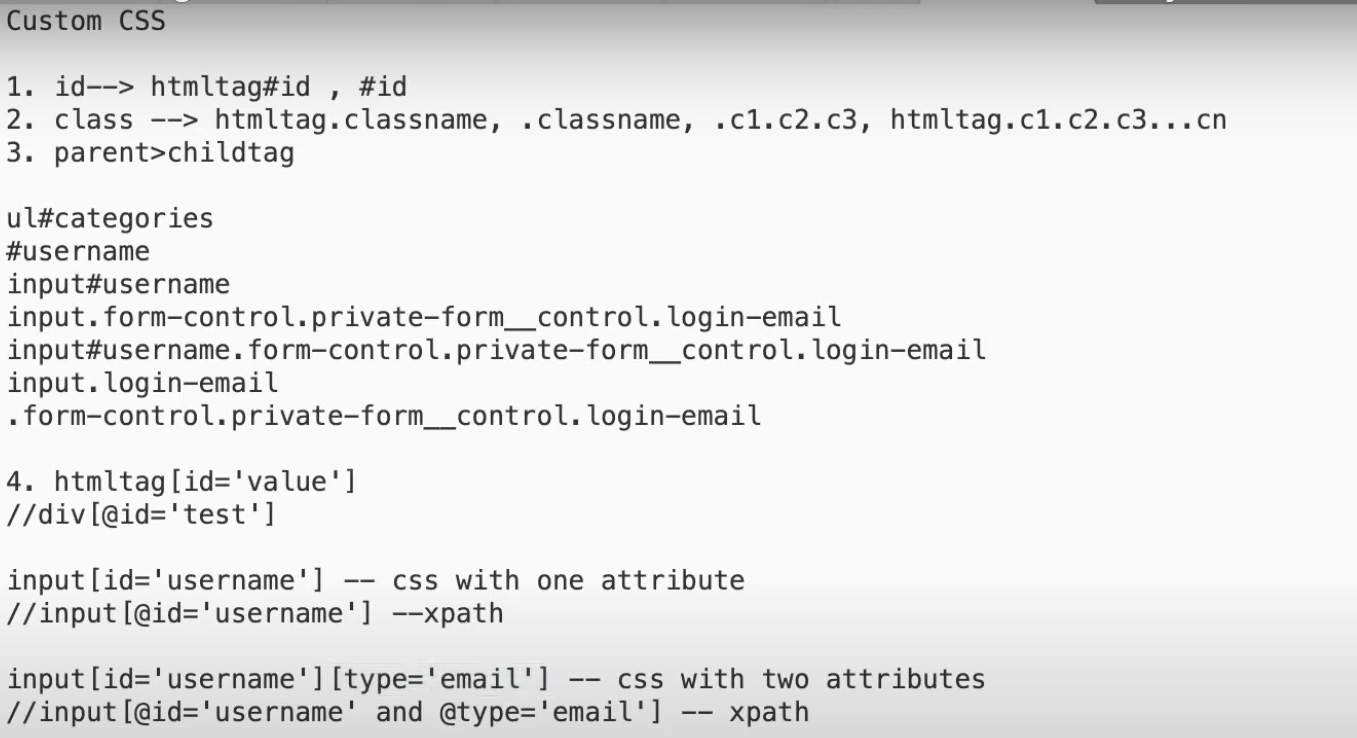
------------

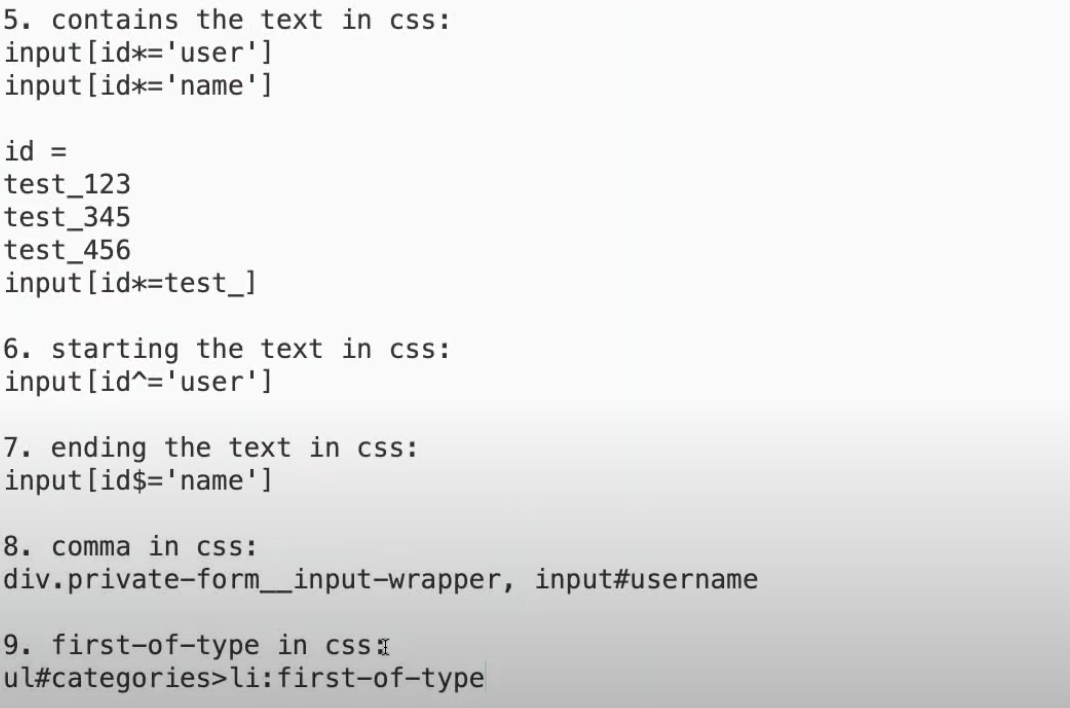
tag id --> tag#id

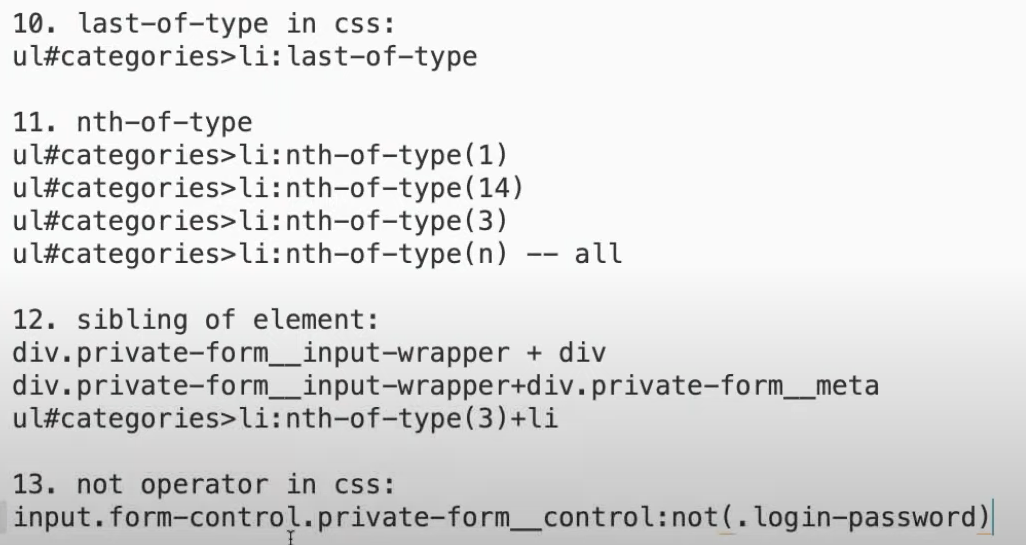
tag class --> tag.class

tag attribute ---> tag[attribute=value]

tag class attribute ---> tag.class[attribute=value]







here is the list of all CSS locators discussed in video:

Custom CSS

1. id--> htmltag#id , #id

2. class --> htmltag.classname, .classname, .c1.c2.c3, htmltag.c1.c2.c3...cn

3. parent>childtag

|  |
| --- |
|  |
| 1.ul#categories |
| #username |
| input#username |
| input.form-control.private-form\_\_control.login-email |
| 2input#username.form-control.private-form\_\_control.login-email |
| input.login-email |
| .form-control.private-form\_\_control.login-email |
|  |
| 4. htmltag[id='value'] |
| //div[@id='test'] |
|  |
| input[id='username'] -- css with one attribute |
| //input[@id='username'] –xpath |
|  |
| input[id='username'][type='email'] -- css with two attributes |
| //input[@id='username' and @type='email'] -- xpath |
|  |
| 5. contains the text in css: |
| input[id\*='user'] |
| input[id\*='name'] |
|  |
| id = |
| test\_123 |
| test\_345 |
| test\_456 |
| input[id\*=test\_] |
|  |
| 6. starting the text in css: |
| input[id^='user'] |
|  |
| 7. ending the text in css: |
| input[id$='name'] |
|  |
| 8. comma in css: |
| div.private-form\_\_input-wrapper, input#username |
|  |
| 9. first-of-type in css: |
| ul#categories>li:first-of-type |
|  |
| 10. last-of-type in css: |
| ul#categories>li:last-of-type |
|  |
| 11. nth-of-type |
| ul#categories>li:nth-of-type(1) |
| ul#categories>li:nth-of-type(14) |
| ul#categories>li:nth-of-type(3) |
| ul#categories>li:nth-of-type(n) -- all |
|  |
| 12. sibling of element: |
| div.private-form\_\_input-wrapper + div |
| div.private-form\_\_input-wrapper+div.private-form\_\_meta |
| ul#categories>li:nth-of-type(3)+li |
|  |
| 13. not operator in css: |
| input.form-control.private-form\_\_control:not(.login-password) |

<https://guide.blazemeter.com/hc/en-us>

|  |
| --- |
| ul#categories 🡪 by id  htmltag#ID |
| ul#categories>li>a>span.red\_trangle 🡺 15 items selected  htmlTag#id>htmltag>htmltag>htmltag.class |
| ul#categories>li:nth-of-type(2)>a.parent>span.red\_trangle:last-of-type  starts with htmltag  # 🡪 id  > 🡪 next htmltag  : 🡪 specifies the nth element  > 🡪 next htmltag  . 🡪 class  > 🡪 next htmltag  . 🡪 class  : 🡪 last element |
| ul#categories>li:first-of-type 🡪fist element  ul#categories>li:last-of-type 🡪last element  ul#categories>li:nth-of-type(5) 🡪 nth element  ul#categories>li:nth-of-type(n) 🡪 all elements |
| Sibling of element  div.private-form\_\_input-wrapper+div  htmltag.class+htmltag  next sibling of the specified element  ul#categories>li:nth-of-type(5)+li |
| <https://app.hubspot.com/login>  to capture username and password  class = "form-control private-form\_\_control login-email" ---> email element  class = "form-control private-form\_\_control login-password m-bottom-3" ==-> password element  Select the username  Class=”form-control private-form\_\_control login-email” 🡪 this element has 3 classes separated by space 🡪 while getting css replace space with “.”  input.form-control.private-form\_\_control.login-email  Not operator 🡪this select the password  input.form-control.private-form\_\_control:not(.login-email) |
|  |

Interview quest : to fint the custom CSS path