**Basic Locators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Locator Type** | **Syntax** | **Example** | **Selenium syntax Example** |
| Xpath | //tagname[attribute=value] | "//input[**@**name='email']" | driver.find\_element\_by\_xpath("//input[@name='email']") |
| Css selector | tagname[attribute=value] | "input[name='email']" | driver.find\_element\_by\_css\_selector("input[name='email']") |
| ID | No syntax | "id" | driver.find\_element\_by\_id("exampleFormControlSelect1") |
| Name | No syntax | "name" | driver.find\_element\_by\_name("name") |
| Class name | No syntax | "class-name" | driver.find\_element\_by\_class\_name("btn-success") |
| Link text | No syntax | "link-text" | driver.find\_element\_by\_link\_text("Genealogies") |
| Partial link text | No syntax | "partialtext" | driver.find\_element\_by\_partial\_link\_text("partialtext") |
| Tag name | No syntax | "tag-name" | driver.find\_element\_by5\_tag\_name("span") |

**Customized XPATH Locator Without Tag Name**

Replace tag name with asterisk (\*) sign

***Syntax:*** //\*[attribute=value]

***Example***: //\*[@name = ‘email’]

**Customized CSS Selector Syntax Without Tag Name**

Remove tag name

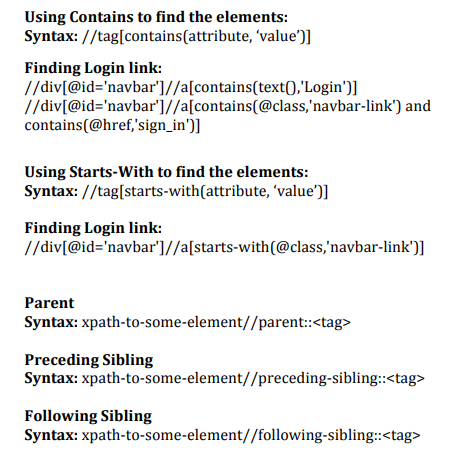
***Syntax***: [attribute=value]

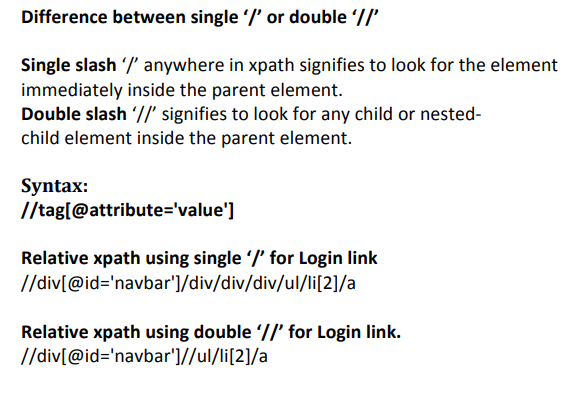
***Example***: [class\*=‘alert-succes’]

**Generating XPATH based on text**

***Syntax:***  //tagname[contains(text(), ‘actual-text’)]

***Example***: //span[contains(text(),'Users Info')]





## Creating XPATH by traversing tags

***Syntax:*** ParentTag/ChildTag

***Example:*** //div[@class='product-action']/button

## Creating a CSS Selector by traversing to nth child

***Syntax***: Tagname:nth-child(x)

***Example***: div:nth-child(1)

## Select Parent Locator from Child using XPATH

***Syntax*** XPATH/parent::tagname

***Example***: //\*[title="test"]/parent::div

## Generating CSS Selector from Tag and Class Name

Replace spaces with period(.) to use more than one class name

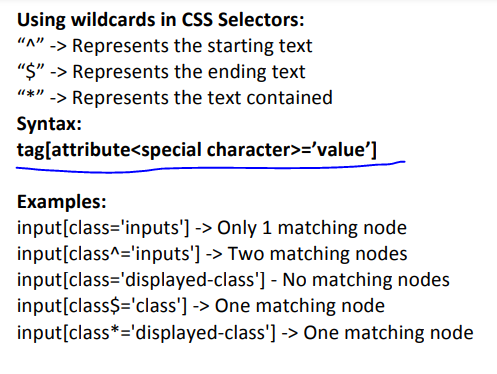
***Syntax***: Tagname.ClassName

***Example***: input.search-keyword

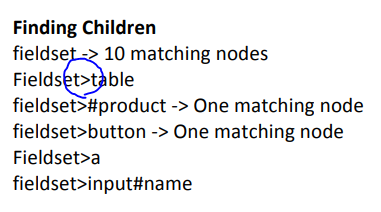


## Generating CSS from ID:

Tagname#ID—Tagname optional



* We use right arrow( >) to traverse to child in css where as in xpath we use **/**





## There are five types of CSS Selectors available, those are:

1) Simple Selectors

2) Combinator selectors

3) Pseudo-class selectors

4) Pseudo-elements selectors

5) Attribute selectors

-- Simple Selectors --

Used for finding the elements based on simple attributes (id and class)

-- Combinator selectors --

Used for defining the relationship between elements in the HTML webpage.

-- Pseudo-class selectors --

Used for finding the elements based on element locations/positions/indexes

-- Pseudo-elements selectors --

Used for selecting an element then apply a style to part of that element.

-- Attribute selectors --

Used for finding the elements based on attributes and attribute values

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

CSS Selector is a pattern used to find the elements in HTML web page.

If you are writing that pattern using simple attributes like id and class, then we call that pattern as CSS Simple Pattern.

There are four patterns available in CSS Simple Selectors.

1) CSS Element Selector

2) CSS ID Selector

3) CSS Class Selector

4) CSS Universal Selector

CSS Element Selector:- This finds the elements in the HTML page based on the element tagname.

CSS ID Selector:- This finds the elements in the HTML page based on the element's id attribute value. **#ID**

CSS Class Selector:- This finds the elements in the HTML page based on the element's class attribute value. .Classattribute

CSS Universal Selector:- The universal selector (\*) **finds all HTML elements on the page**. \*

If you are writing that pattern using attributes and attribute values, then we call that pattern as CSS Attribute Pattern.

The [attribute] selector is used to find elements with a specified attribute.

The [attribute="value"] selector is used to find elements with a specified attribute and value.

The [attribute~="value"] selector is used to find elements with an attribute value containing a specified word.

The [attribute\*="value"] selector is used to find elements whose attribute value contains a specified value.

The [attribute|="value"] selector is used to find elements with the specified attribute starting with the specified value.

The [attribute^="value"] selector is used to find elements whose attribute value begins with a specified value.

The [attribute$="value"] selector is used to find elements whose attribute value ends with a specified value.

Examples:

AttributeName ------------------------ input[placeholder]

AttributeValue ----------------------- input[placeholder='First Name']

AttributeValue-PartialText-WholeWord - input[placeholder~='question']

AttributeValue-PartialText-Text ------ input[placeholder\*='que']

AttributeValue-StartsWith-WholeWord -- p[class|='my'] ->splits word with dash(-)

AttributeValue-StartsWith-Text ------- p[class^='my']

AttributeValue-EndsWith-Text --------- p[class$='lass']

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

In this video, I explained about CSS Combinator Selectors.

CSS Selector is a pattern used to find the elements in the HTML web page.

If you are not able to find any element in the webpage uniquely or if the element is not having any attributes then, in that case, we are going to use relationships means using the reference of other elements and find the current element.

A combinator is something that explains the relationship between the selectors.

There are four different combinator selectors available in CSS:

➳ descendant selector

➳ child selector

➳ adjacent sibling selector

➳ general sibling selector

⭐ Descendant Selector ⭐

The descendant selector matches all elements that are descendants of a specified element.Give space to find these

⭐ Child Selector ⭐ **>**

The child selector selects all elements that are the children of a specified element.

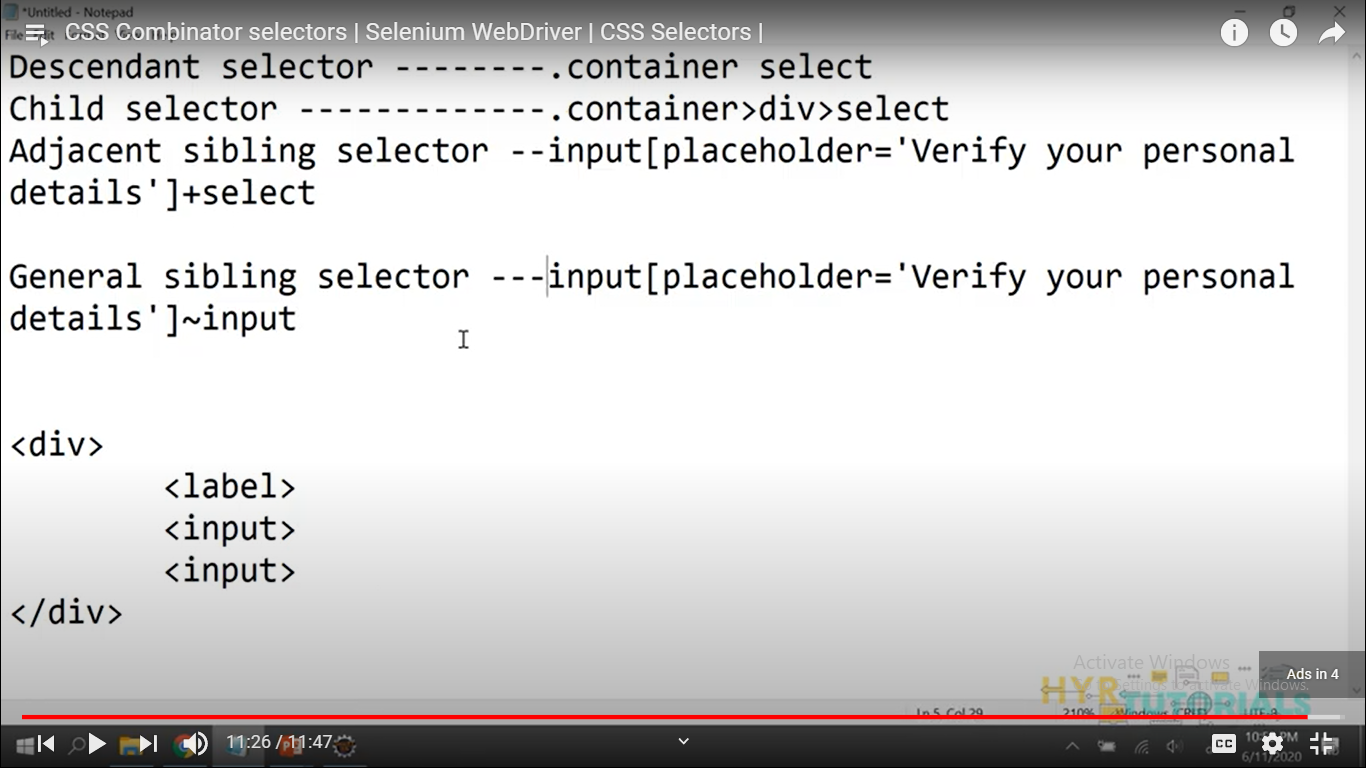
⭐ Adjacent Sibling Selector ⭐ +

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

⭐ General Sibling Selector ⭐ **~**

The general sibling selector selects all elements that are siblings of a specified element.



In this video, I explained about Conditions and Pseudo Classes in CSS Selectors.

CSS Selector is a pattern used to find the elements in the HTML web page.

If you are not able to find any element in the webpage uniquely by just using single attribute value and relations then we have to take the help of conditions and pseudo-classes.

⭐ Conditions in CSS Selectors ⭐

➳ And ex: input[type='button'][class='button']

➳ Or ex: button,input[type='button'],[class='submit']

➳ Not ex: input.button:not([type='button'])

⭐ Pseudo Classes in CSS Selectors ⭐

➳ first-child

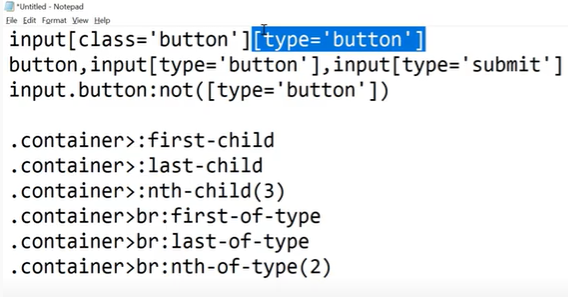
➳ last-child

➳ nth-child

➳ first-of-type (filter based on tagname etc)

➳ last-of-type

➳ nth-of-type



## ------- XPath Operators -------

= Tests whether two expressions are equal.

＜ Tests whether the first expression is less than the second.

＜= Tests whether the first expression is less than or equal to the second.

＞ Tests whether the first expression is greater than the second.

＞= Tests whether the first expression is greater than or equal to the second.

!= Tests whether the two expressions are not equal.

and - Tests whether both the first and second expressions are true. If the first expression is false, the second is not evaluated.

or - Tests whether either the first or second expressions are true. If the first expression is true, the second is not evaluated

------- XPath Functions -------

**text()** - Selects the text nodes of the current context node.

**Contains()** - Contains method is used when we know about the partial attribute value or partial text associated with the web element.

**Starts-with()** - starts-with method is used when we know about the initial partial attribute value or initial partial text associated with the web element.

**normalize-space()** - The normalize-space function ignores the leading, trailing, and repeating white spaces, that means after applying the normalize-space the text become normalized with no line breaks, and give a proper sentence.

------- Examples -------

//td[text()=5000]

//button[@id='refreshBtn' and @value='Refresh']

//button[@id='refreshBtn' or @value='Refresh']

//a[text()='Sign in']

//a[contains(text(),'Sign')]

//button[contains(@class,'sign')]

//div[starts-with(@class,'container')]

//label[normalize-space(text())='First Name']

------ Axes names/methods -------

👉 ancestor

👉 ancestor-or-self

👉 descendant

👉 descendant-or-self

👉 preceding-sibling

👉 following-sibling

👉 parent

👉 child

👉 following

👉 preceding

------ Examples ------

//label[text()='Email']/following-sibling::input[1]

//td[text()='Maria Anders']/preceding-sibling::td/child::input

//label[text()='Email']/following-sibling::input[1]/parent::div

//div[@class='container']/child::input[@type='text']

//div[@class='container']/descendant::button

//div[@class='buttons']/ancestor-or-self::div

//label[text()='Password']/following::input[1]

Other things:

* Tagname ‘**a’** is for links.

print(driver.current\_url())

driver.title()

driver.refresh()

driver.close()

driver.back() to go back to previous page

driver.maximise\_widow()

