Types of CSS Selectors in Selenium & it's usages

1) In Selenium we can find element on the webpage using different locators, CSS Selector is also one of the locator which we can use to identify the element on the webpage.
2) When we are not able to identify a element on the webpage using locators like Id, Name, ClassName, Tagname, PartialLinkText & LinkText. We go for XPath and CSS Selectors where we can combine multiple attributes and functions to identify a element uniquely.
3) CSS Selectors are faster compared to xpath and are more readable.
4) While using CSS Selector we can only identify elements moving in the forward direction, we cannot traverse back in DOM to locate a element.
5) CSS Selectors locator will directly hit the node we want to locate instead of traversing through the whole DOM.
6) We can create our own custom CSS Selectors using combinations of id, class, AttributeName, Conditions & function.
7) For element having id we use '#' (hash) sign.
8) For element having class name we use '.' (dot) sign.
9) We do not use '//' double forward slash at the start of css selector locator, instead html tag name or simple id or class name can be used.
10) Syntax:
    i. ID: *#idName*
    ii. ClassName: *.className*
    iii. HTML tag with id: *htmltag#idName*
    iv. HTML tag with className: *htmltag.className*
    v. Combination of above all: *htmltag#idName.className*

    Note: There can be various combinations which can be generated by using id, class name, attribute name, conditions and functions. The above are just few basic example.

➢ Different options available to create Custom CSS Selectors locator:

1) ID
2) Class Name
3) HTML tag with ID
4) HTML tag with class name
5) Attribute name = 'value'
6) AND Condition
7) OR Condition
8) NOT Condition
9) Starts-with function
10) Ends-with function
11) Contains function
12) Descendant relationship
13) Parent Child relationship
14) Siblings relationships
15) first-child function
16) last–child function
17) nth-child function
18) first-of-type function
19) last-of-type function
20) nth-of-type function
21) Combinations of above all

➢ HTML DOM Element:

<button class="btn_primary" id="add-to-cart" name="backpack">Add to cart one</button>

<button class="btn btn_primary btn_small btn_inventory" id="backpack" name="sauce-labs" data-test="cart-button">Add to cart two</button>

➢ CSS Selectors for above HTML DOM Element:

1) Using just ID: *#add-to-cart*
2) Using just class name: *.btn_inventory*
3) Using HTML tag with ID: *button#add-to-cart*
4) Using HTML tag with class name: *button.btn_inventory*
5) Using HTML tag with ID & class name: *button#add-to-cart.btn_inventory*
6) Using multiple class names: *.btn.btn_primary.btn_small.btn_inventory* (remove all the whitespace between calss name and add a dot between each class name)
7) Using multiple class with an ID : *.btn.btn_primary.btn_small#backpack*
8) Using HTML tag with multiple class name : *button .btn.btn_primary.btn_small.btn_inventory*
9) Using class name with attribute name and value: *.btn.btn_primary[name='sauce-labs']*
10) Using ID with attribute name and value : *#backpack[name='sauce-labs']*
11) Using HTML tag with ID and attribute name & value: *button#backpack[name='sauce-labs']*
12) Using HTML tag with class name and attribute name and value: *button.btn.btn_primary[name='sauce-labs']*
13) Using HTML tag along with multiple attribute name and value to identify the element (equivalent to AND Condition): *button[id='backpack'][name='sauce-labs']*

    No need to use AND in the locator instead separate multiple attributes using '[]' square brackets each attribute will be added in separate '[]' square brackets.
14) Using HTML tag with ID along with multiple attribute name and value to identify the element: *button#backpack[name='sauce-labs'][data-test='cart-button']*
15) Using HTML tag with class name along with multiple attribute name and value to identify the element: *button.btn_primary[name='sauce-labs'][data-test='cart-button']*
16) Using multiple HTML tag along to identify at least one element from them(equivalent to OR Condition): *button[name='sauce-labs'],input[type='button'],input[type='submit']*

    No need to use OR in the locator instead separate multiple HTML tag using ',' comma
17) Using HTML tag along with multiple attribute name and excluding some attribute to locate a element uniquely (NOT Condition): *button.btn_primary:not([name='backpack'])*

    The above CSS selector will select both the button using *button.btn_primary* then we are excluding the first button using not condition :*not([name='backpack'])* with attribute name.

➢ HTML DOM Element:

```
<div class="container">
        <input class="name" id="firstName" name="fname" placeholder="First Name">
        <input class="name" id="lastName" name="lname" placeholder="Last Name">
        <input class="gender" placeholder="Gender" type="text">
        <input placeholder="Enter your security question" type="text">
        <input placeholder="“Answer for the question" type="text">
        <div>
                        <button id="button3">Button3</button>
                        <input class="button" type="button" value="Button1">
                        <input class="button" type="submit" value="Button2">
         </div>
        <div class="second">The second div element.</div>
        <div class="my-test">The third div element.</div>
</div>
```

➢ CSS Selectors for above HTML DOM Element:

1) Using starts-with function: *input[type='text'][placeholder^='Answer']* (starts-with function is represented using ‘^’ sign)
2) Using ends-with function: *input[placeholder$='der']* (ends-with functions is represented using ‘$’ sign)
3) Using contains function: *div[class*='test']* (contains function is represented using ‘*’ sign)
4) Using combinations of above all functions: *input[class*='name'][name$='name'][placeholder^='First']*
5) Using Descendant relationship: *div.container button#button3* (to get a descendant of any HTML tag separate the parent and child/grandchild HTML tag with whitespace, the CSS selector will identify button with id = button3 which is a grandchild of main div tag)
6) Using Parent Child relationship: *div>input.button[value='Button2']* (to get the child of any parent HTML tag use ‘>’ sign after parent HTML tag to get all its child tag and then mentioned the required HTML child tag to identify the element, the CSS selector will identify the 2nd input field with value Button2 which is in parent div tag)
7) Using Adjacent sibling: *div>button#button3+input* (to get an adjacent sibling of any HTML tag use ‘+’ sign it will fetch the immediate next sibling only, the CSS Selector will identify the input tag whose value = Button1)
8) Using General sibling: *div>button#button3~input[value='Button2']* (to get specific sibling of any HTML tag use ‘~’ sign it will fetch all the sibling and further we can provide some unique attribute to identify required element, the CSS Selector will identify the input tag whose value = Button2)

Note: Adjacent sibling(+) will fetch the immediate sibling after the specified HTML tag and General Sibling(~) will fetch all the siblings of HTML tag.

➢ HTML DOM Element:

```
<div class="container">
        <input placeholder=“Answer for the question" type="text">
        <div>
                        <button id="button3">Button3</button>
                        <input class="button" type="button" value="Button1">
                        <input class="button" type="submit" value="Button2">
                        <label id="firstName" value="First Name"> First Name</label>
                        <label id="lastName" value="Last Name"> Last Name </label>
                        <label id=“titleName" value=“Title Name"> Title Name </label>
                        <p> This is paragraph </p>
                        <h1> Heading tag </h1>

         </div>
        <div class="second">The second div element.</div>
</div>
```

➢ CSS Selectors for above HTML DOM Element:

1) Using first-child relationship function: *div.container>div>:first-child* (the CSS Selector will fetch the first child element ‘button’ HTML tag of div tag which will be child of main div tag)
2) Using last-child relationship function: *div.container>div>:last-child* (the CSS Selector will identify the last child element ‘h1’ HTML tag of div tag which will be child of main div tag)
3) Using nth-child relationship function with position value: *div.container>div>:nth-child(2)* (the CSS Selector will identify the nth child element with index position 2 ‘input’ HTML tag of div tag which will be child of main div tag)
4) Using first-of-type relationship function: *div.container>div>label:first-of-type* (the CSS Selector will fetch the first child of HTML element type ‘label’ present under div tag which will be child of main div tag we are specifically mentioning the element type we want to fetch along with first-of-type function)
5) Using last-of-type relationship function: *div.container>div>label:last-of-type* (the CSS Selector will fetch the last child of HTML element type ‘label’ present under div tag which will be child of main div tag we are specifically mentioning the element type we want to fetch along with last-of-type function)
6) Using nth-of-type relationship function: *div.container>div>label:nth-of-type(2)* (the CSS Selector will fetch the 2nd child of HTML element type ‘label’ present under div tag which will be child of main div tag we are specifically mentioning the element type we want to fetch along with nth-of-type function)

Note: The main difference between first-child and first-of-type is in first-child we are not mentioning the HTML tag name it will give us whatever element is the first child but in first-of-type we are specially mentioning the tag name we want it will give us the first element by tag name regardless of order.