Locators in selenium are used to find and match the elements of your page that it needs to interact with. There are 8 locators strategies included in Selenium:

* **Identifier**- works with the **id** and **name** attributes of your html tags
* **Id** - The Id strategy looks for an element in the page having an **id** attribute corresponding to the specified pattern. **<label id="my\_id" />** will be matched by a locator like **id=my\_id** or just **my\_id**
* **Name-** The Id strategy looks for an element in the page having an **id** attribute corresponding to the specified pattern. **<label id="my\_id" />** will be matched by a locator like **id=my\_id** or just **my\_id**
* **Link-** This strategy is intended to select links only and selects the anchor element containing the specified text
* **DOM-** The DOM strategy works by locating elements that matches the javascript expression refering to an element in the DOM of the page
* **XPath-**  XPath is the standard navigation tool for XML; and an HTML document is also an XML document (xHTML). XPath is used everywhere where there is XML
* **CSS-** The CSS locator strategy uses CSS selectors to find the elements in the page.
* **UI-element-** UI-element is a rather new locator.It was at first a Selenium IDE extension.It is now fully integrated into Selenium
* Locators can be classified into two categories:
* Structure-based locators: locators that rely on the structure of the page to find elements.
  + XPath
  + DOM
  + CSS
* Attributes-based locators: locators that relies on the attributes of the elements to locate them
  + Identifier
  + Id
  + Name
  + Link
  + CSS

### Identifier

works with the **id** and **name** attributes of your html tags. Let’s consider the following example:

<html>

<body>

<form id="login">

<input name="username" type="text"/>

<input name="password" type="password"/>

<input name="submit" type="submit" value="Continue!"/>

</form>

</body>

</html>

Valid locators for this snippet are :

* **identifier=login**
* **identifier=username**
* **submit**

### Id

The Id strategy looks for an element in the page having an **id** attribute corresponding to the specified pattern. **<label id="my\_id" />** will be matched by a locator like **id=my\_id** or just **my\_id**

**PROS**:

* Each id is supposed to be unique so no chance of matching several elements

**CONS**:

* Works well only on elements with fixed ids and not generated ones

### Name

Like the Id strategy, but on the **name** attribute. You can also specify a filter to refine your locator. Currently, there are two filter types :

* **Value** : matches elements with a **name** attribute and where the **value** follows a pattern. The following example illustrates the interest of filters :
* <html>
* <body>
* <div id="pancakes">
* <button type="button" name="pancake" value="Blueberry">Blueberry</button>
* <button type="button" name="pancake" value="Banana">Banana</button>
* <button type="button" name="pancake" value="Strawberry">Strawberry</button>
* </div>
* </body>

</html>

Scenario:

we just added a strawberry pancake in our application and we want to test that the button that adds it into the cart works. With a locator like **name=pancake**, Selenium will find 3 elements and return the first one : the test will never fail even if the strawberry button is not here! Use a value filter like **name=pancake value=Strawberry** and the locator successfully identifies the Strawberry button.

* **Index** : same as name but works with an index. Using the previous example, the locator **name=pancake index=2** will select the Strawberry button.

|  |  |
| --- | --- |
|  | **Tip** |
| the index starts at 0 |

**PROS**:

* Works well with fixed list of similar elements

**CONS**:

* Difficult to use with data-bound lists

### Link

This strategy is intended to select links only and selects the anchor element containing the specified text: **link=The text of the link**

**PROS**:

* Will only select anchor elements
* Useful when testing navigation

**CONS**:

* You have to know the text of the link before

### 6.5. DOM

The DOM strategy works by locating elements that matches the javascript expression refering to an element in the DOM of the page.

* **dom=document.div['pancakes'].button[0]**
* **document.div[0].button[2]**
* **dom=function foo() { return document.getElementById("pancakes"); }; foo();**

**PROS**:

* Javascript allows you to build dynamic locators

**CONS**:

* Relies on the structure of the page

### 6.6. XPath

While DOM is the recognized standard for navigation through an HTML element tree, XPath is the standard navigation tool for XML; and an HTML document is also an XML document (xHTML). XPath is used everywhere where there is XML. Valid XPath locators can be:

* **xpath=//button[@value="Blueberry"]**: matches the Blueberry button
* **//div[@id="pancakes"]/button[0]**: same thing

**PROS**:

* Allows very precise locators

**CONS**:

* Slower than CSS
* Relies on browser’s XPath implementation which is not always complete (especially on IE) and as such is not recommended for cross-browser testing

### 6.7. CSS

The CSS locator strategy uses CSS selectors to find the elements in the page. Selenium supports CSS 1 through 3 selectors syntax excepted CSS3 namespaces and the following:

| **pseudo-classes** | **pseudo-elements** |
| --- | --- |
| **:nth-of-type** | **::first-line** |
| **:nth-last-of-type** | **::first-letter** |
| **:first-of-type** | **::selection** |
| **:last-of-type** | **::before** |
| **:only-of-type** | **::after** |
| **:visited** |  |
| **:hover** |  |
| **:active** |  |
| **:focus** |  |
| **:indeterminate** |  |

* **css=div[id="pancakes"] > button[value="Blueberry"]** selects the button with its value property set at Blueberry if children of the pancakes div

**PROS**:

* Much faster than XPath
* Widely used
* Provides a good balance between structure and attributes
* Allows for selection of elements by their surrounding context

**CONS**:

* They tend to be more complex and require a steeper learning curve

### 6.8. UI-Elements

* UI-element is a rather new locator
* It was at first a Selenium IDE extension
* It is now fully integrated into Selenium
* See the [Section 9.4, “UI-Elements:”](https://www.protechtraining.com/bookshelf/selenium_tutorial/testing_strategies#UI-Elements)

|  |  |
| --- | --- |
|  | **Caution** |
| As a general rule, keep in mind that if a locator matches several elements, only the first one will be effectively used by Selenium |

### 6.9. Structure-Dependent Or Not?

* Locators can be classified into two categories:
  + Structure-based locators: locators that rely on the structure of the page to find elements.
    - XPath
    - DOM
    - CSS
  + Attributes-based locators: locators that relies on the attributes of the elements to locate them
    - Identifier
    - Id
    - Name
    - Link
    - CSS
* You should consider this before choosing a locator strategy
* Most people choose CSS because it is the most flexible and gives a good balance between using structure and attributes to find the elements