

CSS Selectors Reference Card

The CSS locator strategy in Selenium is now powered by the Sizzle JS library. Sizzle implements most of the W3C CSS3 selectors, though some will rarely be used in the content of Selenium.

The ideal situation is to reference items on the page by their unique id. If that is not possible, then XPath or CSS are the strategies available to you. XPath however has performance issues in Internet Explorer. Converting your XPath to CSS has shown a considerable decrease in script execution duration.

Location strategies can be mixed-and-matched depending on need. Converting all XPath to CSS may not be valuable unless scripts are taking too long to run.

There is not a direct mapping of functionality between XPath and CSS. For example, there is no equivalent to `getXpathCount`.

Like XPath, the CSS strategy is a structurally based one so it is highly recommended that you make your CSS selectors as specific as possible lest they become too sensitive to changes in the page.

All CSS locators must be preceded with `css=` to tell Selenium that this is a CSS locator and not an id.

Basics

Element

- `css=input`

Element id

- `css=input#inputUser`

Element class

- `css=input.plain`

Relativity

Child

- `css=form > input.formSubmit`

Descendant

- `css=body input#inputUse`

Contiguous

- `css=input#inputUser + input#inputPass`

Shared Parent

- `css=input#inputUser ~ input#inputPass`

Position

First

- `css=ul#reasons li:first`

Last

- `css=ul#reasons li:last`

Specific

- `css=ul#reasons li:nth(2)`

Attributes

Single

- `css=input[value=lacrosse]`

Chained

- `css=input[value=swim][type=checkbox]`

Starts-with

- `css=input[value^=ballroom]`

End-with

- `css=input[value$=field]`

Contains

- `css=input[value*=room]`

Negatives

Not

- `css=input[value$=d]:not([value^=b])`

Text

Contains

- `css=input:contains(lacrosse)`



<http://saucelabs.com>