## **CSS to XPath Conversion Guide**

### **Targeting using Element Tags**

| Description | CSS | XPath |
| --- | --- | --- |
| Any element | \* | //\* |
| Any <p> element | p | //p |
| Any <div> element | div | //div |

### **Targeting using Attributes**

| Description | CSS | XPath |
| --- | --- | --- |
| Any <button> where name equals send | button[name="send"] | //button[@name="send"] |
| Any element where type equals submit | [type="submit"] | //\*[@type="submit"] |
| Any element where href begins with abc | [href^="abc"] | //\*[starts-with(@href, "abc")] |
| Any element where href ends with xyz | [href$="abc"] | //\*[ends-with(@href, "abc")] |
| Any element where href contains mno | [href\*="mno"] | //\*[contains(@href, "abc")] |

**Targeting using IDs**

| Description | CSS | XPath |
| --- | --- | --- |
| Any element where id equals sample | #sample | //\*[@id="sample"] |

**Targeting using Classes**

| Description | CSS | XPath |
| --- | --- | --- |
| Any element where class includes highlight | .highlight | //\*[contains(concat(" ", @class, " "), " highlight ")] |
| Any element where class includes highlight | .highlight | //\*[@class="highlight")] |

### **Targeting using Contents**

| Description | CSS | XPath |
| --- | --- | --- |
| Any h1 element containing text Dashboard | h1:contains("Dashboard") | //h1[contains(text(), "Dashboard")] |

**Targeting using Hierarchy**

Leveraging hierarchy allows us to match descendent elements that are nested inside of other elements. Here's some sample HTML to help clarify the difference between a "child" and a "descendent".

<div id="sample">

<span>This is both a child and descendent of #sample.</span>

<div class="nested">

<span>This is a descendent of #sample only.</span>

</div>

</div>

| Description | CSS | XPath |
| --- | --- | --- |
| Any <span> element that's a "direct" child of #sample | #sample > span | //\*[@id="sample"]/span |
| Any <span> element that's a descendent of #sample | #sample span | //\*[@id="sample"]//span |

**Additional Targeting Options**

| Description | CSS | XPath |
| --- | --- | --- |
| The 3rd <li> element in a list | li:nth-of-type(3) | //li[3] |
| The last <li> element in a list | li:last-child | //li[last()] |

**Trim/Remove Spaces**

The normalize-space function strips leading and trailing white-space from a string, replaces sequences of white-space characters by a single space, and returns the resulting string.

Example : Which one is better?

//td[starts-with(normalize-space(),'Text to Trim')]

//td[starts-with(normalize-space(text()),'Text to Trim')]

The second option is better as it targets at the specific nodes

More string functions in XPATH:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Functions | Description | | starts-with(*string1*, *string2*) | Returns true if the first string starts with the second string. | | contains(*string1*, *string2*) | Returns true if the first string contains the second string. | | substring(*stringoffsetlength*) | Returns a section of the string. The section starts at *offset* (which is a number), and is as long as the value provided at *length.* | | substring-before(*string1*, *string2*) | Returns the part of *string1* up until the first occurence of *string2*. | | substring-after(*string1*, *string2*) | Returns the part of *string1* after the first occurence of *string2*. | | string-length(*string*) | Returns the length of *string* (i.e. the number of characters). | | normalize-space(*string*) | Trims the leading and trailing space from *string*. Also replaces consecutive occurrences of white space with a single space. | | translate(*string1*, *string2*, *string3*) | Returns *string1* after any matching characters in *string2* have been replaced by the characters in *string3*. | | concat(*string1*, *string2*, *...*) | Concatenates all strings (i.e. joins them together). | | format-number(*number1*, *string1*,*string2*) | Returns a formatted string version of *number1* after applying *string1* as a format string. *string2* is an optional locale string. | |  |

|  |  |
| --- | --- |
| Document information | |
| document-uri(node) | a string describing the URI of node |
|  |  |
| Mathematics | |
| abs(numeric) | absolute value of numeric |
| ceiling(numeric) | smallest integer greater than numeric |
| floor(numeric) | biggest integer smaller than numeric |
|  |  |
| Strings | |
| concat(arg1,arg2 ...) | string concatenating the string values of argi |
| string-join(strings, sep) | string concatenating the members of the sequence strings using the string sep as a separator |
| substring(string, start, length) | string portion with string, from position start (the first character is numbered 1) for length characters; if length is not given, then the rest of the string is returned. |
| string-length(string) | number of characters in string |
|  |  |
| normalize-space(string) | string after removing leading and trailing blanks and replacing sequences of one or more than one whitespace character with a single space (#x20) |
| translate(string, mapStr, transStr) | string in which every character appearing at position N in the mapStr is replaced by the character at position N in the transStr |
| lower-case upper-case (string) | all lower or upper case version of string |
|  |  |
| contains(string1, string2) | checks if string1 contains string2 |
| starts-with(string1, string2) | checks if string1 starts with string2 |
| ends-with(string1, string2) | checks if string1 ends with string2 |
| substring-before(string1, string2) | string of chars of string1 before the first occurrence of string2 |
| substring-after(string1, string2) | string of chars of string1 after the first occurrence of string2 |
|  |  |
| matches(string,pattern) | checks if string matches the regular expression pattern; unless ^ and $ are used as anchors, the string matches if one of its substring does |
| replace(string,pattern,replacement) | string obtained after replacing within string every non-overlapping occurrence of pattern by replacement |
| tokenize(string,pattern) | break string into a sequence of strings, treating every substring matching pattern as a separator |
|  |  |
| Sequences | |
| distinct-values(seq) | sequence of different values that appear at least once in seq |
| remove(seq, position) | sequence of all elements of seq except for the one at position |
| reverse(seq) | sequence of all elements of seq but in reverse order |
| subsequence(seq, start, length) | sequence of elements of seq, from position start (the first element is numbered 1) for length elements; if length is not given, then the rest of the sequence is returned. |
|  |  |
| count(seq) | number of elements within seq |
| avg|max|min|sum(seq) | average | maximum | minimum | sum of the values within seq |
|  |  |
| Context | |
| position() | position of the context item within the current sequence of items |
| last() | the size of the current sequence of items |
| current-date|current-time() | date or time value of now |