

Xpath cheatsheet

Xpath test bed

Test queries in the Xpath test bed:

Xpath test bed

(whitebeam.org)



Browser console

```
$x("//div")
```

Works in Firefox and Chrome.

Selectors

Descendant selectors

h1	//h1	?
div p	//div//p	?
ul > li	//ul/li	?
ul > li > a	//ul/li/a	
div > *	//div/*	
:root	/	?
:root > body	/body	

Attribute selectors

<code>#id</code>	<code>//*[@id="id"]</code>	?
<code>.class</code>	<code>//*[@class="class"] ...kinda</code>	
<code>input[type="submit"]</code>	<code>//input[@type="submit"]</code>	
<code>a#abc[for="xyz"]</code>	<code>//a[@id="abc"][@for="xyz"]</code>	?
<code>a[rel]</code>	<code>//a[@rel]</code>	
<code>a[href^='/']</code>	<code>//a[starts-with(@href, '/')]</code>	?
<code>a[href\$='.pdf']</code>	<code>//a[ends-with(@href, '.pdf')]</code>	
<code>a[href*='://']</code>	<code>//a[contains(@href, '://')]</code>	
<code>a[rel~='help']</code>	<code>//a[contains(@rel, 'help')] ...kinda</code>	

Order selectors

<code>ul > li:first-of-type</code>	<code>//ul/li[1]</code>	?
<code>ul > li:nth-of-type(2)</code>	<code>//ul/li[2]</code>	
<code>ul > li:last-of-type</code>	<code>//ul/li[last()]</code>	
<code>li#id:first-of-type</code>	<code>//li[1][@id="id"]</code>	?
<code>a:first-child</code>	<code>//*[1][name()="a"]</code>	
<code>a:last-child</code>	<code>//*[last()][name()="a"]</code>	

Siblings

<code>h1 ~ ul</code>	<code>//h1/following-sibling::ul</code>	?
<code>h1 + ul</code>	<code>//h1/following-sibling::ul[1]</code>	
<code>h1 ~ #id</code>	<code>//h1/following-sibling::[@id="id"]</code>	

jQuery

<code>\$('#ul > li').parent()</code>	<code>//ul/li/..</code>	?
<code>\$('#li').closest('section')</code>	<code>//li/ancestor-or-self::section</code>	
<code>\$('#a').attr('href')</code>	<code>//a/@href</code>	?
<code>\$('#span').text()</code>	<code>//span/text()</code>	

Other things

<code>h1:not([id])</code>	<code>//h1[not(@id)]</code>	?
Text match	<code>//button[text()="Submit"]</code>	?
Text match (substring)	<code>//button[contains(text(),"Go")]</code>	
Arithmetic	<code>//product[@price > 2.50]</code>	
Has children	<code>//ul[*]</code>	
Has children (specific)	<code>//ul[li]</code>	
Or logic	<code>//a[@name or @href]</code>	?
Union (joins results)	<code>//a //div</code>	?

Class check

```
//div[contains(concat(' ',normalize-space(@class),' '), ' foobar ')]
```

Xpath doesn't have the "check if part of space-separated list" operator, so this is the workaround ([source](#)).

Expressions

Steps and axes

//	ul	/	a[@id='link']
Axis	Step	Axis	Step

Prefixes

Prefix	Example	What
//	//hr[@class='edge']	Anywhere
./	./a	Relative
/	/html/body/div	Root
Begin your expression with any of these.		

Axes

Axis	Example	What
/	//ul/li/a	Child
//	//[@id="list"]//a	Descendant

Separate your steps with /. Use two (//) if you don't want to select direct children.

Steps

```
//div
//div[@name='box']
//[@id='link']
```

A step may have an element name (div) and predicates ([. . .]). Both are optional. They can also be these other things:

```
//a/text()      #=> "Go home"
//a/@href       #=> "index.html"
//a/*          #=> All a's child elements
```

Predicates

Predicates

```
//div[true()]
//div[@class="head"]
//div[@class="head"][@id="top"]
```

Restricts a nodeset only if some condition is true. They can be chained.

Operators

```
# Comparison
//a[@id = "xyz"]
//a[@id != "xyz"]
//a[@price > 25]
```

```
# Logic (and/or)
//div[@id="head" and position()=2]
//div[(x and y) or not(z)]
```

Use comparison and logic operators to make conditionals.

Using nodes

```
# Use them inside functions
//ul[count(li) > 2]
//ul[count(li[@class='hide']) > 0]
```

```
# This returns `<ul>` that has a `<li>` child
//ul[li]
```

You can use nodes inside predicates.

Indexing

```
//a[1]           # first <a>
//a[last()]      # last <a>
//ol/li[2]       # second <li>
//ol/li[position()=2] # same as above
//ol/li[position()>1] # :not(:first-of-type)
```

Use `[]` with a number, or `last()` or `position()`.

Chaining order

```
a[1][@href='/']  
a[@href='/'][1]
```

Order is significant, these two are different.

Nesting predicates

```
//section[.//h1[@id='hi']]
```

This returns <section> if it has an <h1> descendant with id='hi'.

Functions

Node functions

name()	# //[starts-with(name(), 'h')]
text()	# //button[text()='Submit']
	# //button/text()
lang(str)	
namespace-uri()	
count()	# //table[count(tr)=1]
position()	# //ol/li[position()=2]

Boolean functions

```
not(expr)          # button[not(starts-with(text()), "Submit")]
```

String functions

```
contains()           # font[contains(@class,"head")]
starts-with()        # font[starts-with(@class,"head")]
ends-with()          # font[ends-with(@class,"head")]
```

```
concat(x,y)
substring(str, start, len)
substring-before("01/02", "/")  #=> 01
substring-after("01/02", "/")   #=> 02
translate()
normalize-space()
string-length()
```

Type conversion

```
string()
number()
boolean()
```

Axes

Using axes

```
//ul/li           # ul > li
//ul/child::li    # ul > li (same)
//ul/following-sibling::li  # ul ~ li
//ul/descendant-or-self::li  # ul li
//ul/ancestor-or-self::li    # $('ul').closest('li')
```

Steps of an expression are separated by /, usually used to pick child nodes. That's not always true: you can specify a different "axis" with ::.

//	ul	/child::	li
Axis	Step	Axis	Step

Child axis

```
# both the same
//ul/li/a
//child::ul/child::li/child::a
```

child:: is the default axis. This makes //a/b/c work.

```
# both the same
# this works because `child::li` is truthy, so the predicate succeeds
//ul[li]
//ul[child::li]
```

```
# both the same
//ul[count(li) > 2]
//ul[count(child::li) > 2]
```

Descendant-or-self axis

```
# both the same
//div//h4
//div/descendant-or-self::h4
```

// is short for the descendant-or-self:: axis.

```
# both the same
//ul//[last()]
//ul/descendant-or-self::[last()]
```

Other axes

Axis	Abbrev	Notes
ancestor		
ancestor-or-self		
attribute	@	@href is short for attribute::href
child		div is short for child::div
descendant		
descendant-or-self	//	// is short for /descendant-or-self::node()/
namespace		
self	.	. is short for self::node()
parent is short for parent::node()
following		
following-sibling		
preceding		
preceding-sibling		
There are other axes you can use.		

Unions

```
//a | //span
```

Use | to join two expressions.

More examples

Examples

```
/**                # all elements
count(**)         # count all elements
(//h1)[1]/text()  # text of the first h1 heading
//li[span]        # find a <li> with an <span> inside it
                  # ...expands to //li[child::span]
//ul/li/..        # use .. to select a parent
```

Find a parent

```
//section[h1[@id='section-name']]
```

Finds a <section> that directly contains h1#section-name

```
//section[//h1[@id='section-name']]
```

Finds a <section> that contains h1#section-name. (Same as above, but uses descendant-or-self instead of child)

Closest

```
./ancestor-or-self::[@class="box"]
```

Works like jQuery's `$().closest('.box')`.

Attributes

```
//item[@price > 2*@discount]
```

Finds <item> and check its attributes

References

- [Xpath test bed \(whitebeam.org\)](#)



►  **21 Comments** for this cheatsheet. [Write yours!](#)