Xpath cheatsheet

Design and Development tips in your inbox. Every weekday.

ads via Carbon

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

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

Order selectors

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

Siblings

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

jQuery

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

Other things

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

Class check

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

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(),"S
```

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
//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)

▶ **0 Comments** for this cheatsheet. Write yours!

devhints.io / Search 357+ cheatsheets

Over 357 curated cheatsheets, by developers for developers.

Devhints home

Other HTML cheatsheets

Input tag
cheatsheet

HTML meta tags
cheatsheet

Layout thrashing
cheatsheet

Appcache
cheatsheet

Applinks
cheatsheet

HTML
cheatsheet

Top cheatsheets

Elixir
cheatsheet

ES2015+
cheatsheet

React.js
cheatsheet

Vim
cheatsheet

Vimdiff
cheatsheet

Vim scripting
cheatsheet