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 Chro

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`	`//*`
`.class`	`//*` ...k
`input[type="submit"]`	`//i`
`a#abc[for="xyz"]`	`//a` [@f
`a[rel]`	`//a`
`a[href^='/']`	`//a` wit
`a[href$='pdf']`	`//a` '.p
`a[href*='://']`	`//a` ':/
`a[rel~='help']`	`//a` 'he

Order selectors

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

Siblings

`h1 ~ ul`	`//h` sib
`h1 + ul`	`//h` sib

`a:last-child`	`//a[last()]`

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`	?

`h1 ~ #id`	`//h sib`

jQuery

`$('ul > li').parent`	`//`
`$('li').closest('.s`	`//s sel`
`$('a').attr('href')`	`//a`
`$('span').text()`	`//s`

Class check

```
//div[contains(concat(' '),' foobar ')]
```

Xpath doesn't have the "cl list" operator, so this is the

Expressions

Steps and axes

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

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.

Prefixes

Prefix	Example
`//`	`//hr[@clas`
`./`	`./a`
`/`	`/html/body`

Begin your expression with

Steps

```
//div //div[@name='box'
```

A step may have an eleme ([...]). Both are optional. things:

```
//a/text() #=> "Go home
//a/* #=> All a's child
```

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.

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.

Chaining order

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

Order is significant, these two are different.

Operators

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

```
# Logic (and/or) //div[
//div[(x and y) or not(
```

Use comparison and logic

Indexing

```
//a[1] # first <a> //a[
//ol/li[2] # second <li
same as above //ol/li[p
child)
```

Use [] with a number, or

Nesting predicates

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

This returns <section> if it
id='hi'.

Functions

Node functions

```
name() # //[starts-with(name(), 'h')] text() #
```

Boolean functions

```
not(expr) # button[not(
```

```
//button[text()="Submit"] # //button/text()
lang(str) namespace-uri()
```

```
count() # //table[count(tr)=1] position() #
//ol/li[position()=2]
```

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

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()]
```

```
with(text(),"Submit"))]
```

String functions

```
contains() # font[conta
with() # font[starts-wi
with() # font[ends-with
```

```
concat(x,y) substring(s
before("01/02", "/") #=
after("01/02", "/") #=>
space() string-length()
```

Child axis

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

`child::` is the default axis

```
# both the same # this
truthy, so the predicat
//ul[child::li]
```

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

Other axes

Axis
`ancestor`
`ancestor-or-self`
`attribute`

Unions

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

Use | to join two expressions.

Axis
child
descendant
descendant-or-self
namespace
self
parent
following
following-sibling
preceding
preceding-sibling

There are other axes you

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='secti
```

Finds a <section> that dir

```
//section[//h1[@id='sec
```

Finds a <section> that cor
as above, but uses descen

Closest

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

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

Attributes

```
//item[@price > 2*@disc
```

Finds <item> and check its

References

- Xpath test bed (whitebeam.org)