



# Xpath cheatsheet

- Proudly sponsored by -

**Cloud 66 for Rails** Simple Rails  
Deployments to Any Cloud with \$100 Credit

*ethical* ad by CodeFund

## Xpath test bed

## Browser console

Test queries in the Xpath test bed:  
[Xpath test bed](#) (whitebeam.org)

`$x("//div")`

Works in Firefox

## Selectors

### Descendant selectors

### Attribute selectors

<code>h1</code>	<code>//h1</code>	<code>#id</code>
<code>div p</code>	<code>//div//p</code>	<code>.class</code>
<code>ul &gt; li</code>	<code>//ul/li</code>	<code>input[type="text"]</code>
<code>ul &gt; li &gt; a</code>	<code>//ul/li/a</code>	<code>a#abc[for="x"]</code>
<code>div &gt; *</code>	<code>//div/*</code>	<code>a[rel="nofollow"]</code>
<code>:root</code>	<code>/</code>	<code>a[href^='http://']</code>
<code>:root &gt; body</code>	<code>/body</code>	<code>a[href\$='.pdf']</code>

### Order selectors

<code>ul &gt; li:first-child</code>	<code>//ul/li[1]</code>	<code>a[rel~='help']</code>
-------------------------------------	-------------------------	-----------------------------

ul > li:nth-child(2)	//ul/li[2]	Siblings
ul > li:last-child	//ul/li[last()]	h1 ~ ul
li#id:first-child	//li[@id="id"][1]	h1 + ul
a:first-child	//a[1]	h1 ~ #id
a:last-child	//a[last()]	

## Other things

h1:not([id])	//h1[not(@id)]	? os
Text match	//button[text()="Submit"]	? r(
Text match (substring)	//button[contains(text(),"Go")]	te
Arithmetic	//product[@price > 2.50]	
Has children	//ul[*]	
Has children (specific)	//ul[li]	//div[contai
Or logic	//a[@name or @href]	Xpath doesn't f
Union (joins results)	//a   //div	?

# # Expressions

## Steps and axes

//	ul	/	a[@id='link']	Prefix
Axis	Step	Axis	Step	//
				./
				/
Axis	Example			What

Axis

Example

What

Axis	Example	What
/	//ul/li/a	Steps Child
//	//*[@id="list"]//a	<div> <div>//div</div> <div>//div[@name=</div> <div>//[@id='link</div> </div>

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

A step may have other things:

```
//a/text()
//a/@href
//a/*
```

## # Predicates

Predicates

Operators

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

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

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

```
# Logic (and)
//div[@id="h"]
//div[(x and
```

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]
//a[last()]
//ol/li[2]
//ol/li[positi
//ol/li[positi
```

## Chaining order

Use [] with a n

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

Nesting pred

Order is significant, these two are different.

//section[//

This returns <sc

## # Functions

### Node functions

### Boolean func

```
name()          # //[starts-with(name(), 'h')]  
text()          # //button[text()='Submit']  
                # //button/text()  
  
lang(str)  
namespace-uri()
```

not(expr)

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

### String func

contains()  
starts-with()  
ends-with()

### Type conversion

concat(x,y)  
substring(st

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

be  
af  
)  
sp  
gt

## # Axes

### Using axes

### Child axis

```
//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')
```

# both the s  
//ul/li/a  
//child::ul/

child:: is the

Steps of an expression are separated by /, usually used to pick child nodes. That’s not always true: specify a different “axis” with ::.

# both the s  
# this works  
//ul[li]  
//ul[child::

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

# both the s  
//ul[count(1  
//ul[count(c

Descendant-or-self axis

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

Other axes

- Axis
- ancestor
- ancestor-or-
- attribute
- child
- descendant

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

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

Unions

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

Use | to join two expressions.

- o
- parent
- following
- following-si
- preceding
- preceding-si

There are other

## # More examples

### Examples

### Find a parent

```
//*           # 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
```

//section[h1

Finds a &lt;secti

//section[//

### Closest

Finds a <secti  
child)

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

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

### Attributes


//item[@pric

Finds &lt;item&gt; a

## # References

- [Xpath test bed \(whitebeam.org\)](http://whitebeam.org)



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



Over 383 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 emails

cheatsheet ●

### Top cheatsheets

Elixir

cheatsheet ●

ES2015+

cheatsheet ●

React.js

cheatsheet ●

Vimdiff

cheatsheet ●

Vim

cheatsheet ●

Vim scripting

cheatsheet ●