DEVHINTS.IO

Edit

Xpath cheatsheet

Xpath test bed

Test queries in the Xpath test bed:

Xpath test bed
(whitebeam.org)

Browser

```
$x("//d
```

Works in

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`

Attribut

`#id`
`.class`
`input[t`
`a#abc[f`
`a[rel]`
`a[href^`
`a[href$`
`a[href*`
`a[rel~=`

Order selectors

`ul > li:first-of-type`	`//ul/li[1]`

?

Siblings

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	?

//div[c

Xpath d
(source)

Expressions

Steps and axes

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

Axes

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

Prefixes

Prefix	
//	
./	
/	

Steps

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

```
//div
//div[@
//[@id=
```

A step m
these ot

```
//a/tex
//a/@hr
//a/*
```

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]
```

Operatc

```
# Compa
//a[@id
//a[@id
//a[@pr
```

```
# Logic
//div[@
//div[(
```

om

Indexing

```
//a[1]
//a[las
//ol/li
//ol/li
//ol/li
```

Use [] w

Nesting

Order is significant, these two are different.

```
//secti
```

This retu

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]
```

Type conversion

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

Boolean

```
not(exp
```

String fu

```
contain
starts-
ends-wi
```

```
concat(
substri
```

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 can specify a different "axis" with `::`.

Child ax

```
# both
//ul/li
//child
```

```
child::
```

```
# both
# this
```

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

Use | to join two expressions.

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

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

sto

sto

attribu

child

descend

descend

namespa

self

parent

followi

followi

precedi

precedi

There ar

More examples

Examples Find a p

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

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

```
//ul/li/..          # use .. to select a parent
```

```
//item[
```

Finds <i

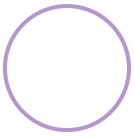
```
//secti
```

Finds a <
instead

References

- Xpath test bed (whitebeam.org)

Search 358+ cheatsheets



Over 358 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