DEVHINTS.IO

Edit

# Xpath cheatsheet

## Xpath test bed

Test queries in the Xpath test bed:

**Xpath test bed**
(whitebeam.org)

## Browser cons

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

Works in Firefo

# 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 sele

| |
|---|
| `#id` |
| `.class` |
| `input[type="` |
| `a#abc[for="x` |
| `a[rel]` |
| `a[href^='/']` |
| `a[href$='pdf` |
| `a[href*='://` |
| `a[rel~='help` |

## 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]` |
| `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` |

### Siblings

| |
|---|
| `h1 ~ ul` |
| `h1 + ul` |
| `h1 ~ #id` |

### jQuery

| |
|---|
| `$('ul > li')` |

### Class check

| |
|---|
| `//div[contai` |
| Xpath doesn't h |

# Expressions

### Steps and axes

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

### Prefixes

| Prefix |
|---|
| `//` |
| `./` |
| `/` |

### Axes

| Axis | Example | What |
|---|---|---|

| Axis | Example | What |
|---|---|---|
| / | //ul/li/a | |
| // | //[@id="list"]//a | |

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

### Steps Child

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

A step may hav
other things:

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

### Operators

```
# Comparison
//a[@id = "x
//a[@id != "
//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[posi
//ol/li[posi
```

### Chaining order

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

Order is significant, these two are different.

### Use [] with a r

### Nesting pred

```
//section[./
```

This returns <s

# 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 func

```
not(expr)
```

### String functic

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

```
concat(x,y)
substring(st
be
af
)
sp
gt
```

### Type conversion

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

# 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')
```

Steps of an expression are separated by `/`, usually used to pick child nodes. That's not always true: 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()]
```

## Unions

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

Use `|` to join two expressions.

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

`child::` is the

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

```
# both the s
//ul[count(l
//ul[count(c
```

## Other axes

| Axis |
|---|
| `ancestor` |
| `ancestor-or-` |
| `attribute` |
| `child` |
| `descendant` |
| `parent` |
| `following` |
| `following-si` |
| `preceding` |
| `preceding-si` |

There are other

# 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
```

Finds a <secti

```
//section[//
```

Finds a <secti child)

## Closest

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

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

## Attributes

```
//item[@pric
```

Finds <item> a

# References

Xpath test bed (whitebeam.org)

**17 Comments** for this cheatsheet. Write yours!

Search 352+ cheatsheets

Over 352 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