THE GREAT CHEATSHEET FOR

# *Xpath*

## 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, '://')]` … *kinda* | |

## 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()]` | |

## 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

Xpath doesn't have the "check if part of space-separated list" operator, so this is the workaround (source):

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

## Expressions

### Prefixes

Begin your expression with any of these.

| | | |
|---|---|---|
| // | *anywhere* | `//hr[@class='edge']` |
| ./ | *relative* | `./a` |
| / | *root* | `/html/body/div` |

### Axes

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

| | | |
|---|---|---|
| / | *child* | `//ul/li/a` |
| // | *descendant* | `//[@id="list"]//a` |

### Steps

A step may have an element name (`div`) and predicates (`[...]`). Both are optional.

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

They can also be these other things.

```
//a/text()          #=> "Go home"
//a/@href           #=> "index.html"
//a/*               #=> All a's child elements
```

## Predicates

### Predicates (`[...]`)

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

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

### Operators
Use comparison and logic operators to make conditionals.

```
# 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)]
```

### Using nodes
You can use nodes inside predicates.

```
# Use them inside functions
  //ul[count(li) > 2]
  //ul[count(li[@class='hide']) > 0]
```

```
# This returns `<ul>` that has a `<li>` child
  //ul[li]
```

### Indexing
Use [ ] with a number, or `last()` or `position()`.

```
//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-child)
```

### Chaining order
Order is significant, these two are different.

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

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

### Nesting predicates

This returns `<section>` if it has an `<h1>` descendant with `id='hi'`.

```
//section[//h1[@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(),"Submit"))]
```

### 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

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/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')
```

### Child axis

This is the default axis. This makes `//a/b/c` work.

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

```
# both the same
# this works because `child::li` is truthy, so the predicate succeeds
  //ul[li]
  //ul[child::li]
```

```
# both the same
  //ul[count(li) > 2]
  //ul[count(child::li) > 2]
```

### Descendant-or-self axis

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

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

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

## Other axes

There are other axes you can use.

| Axis | Abbrev | Description |
|---|---|---|
| `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` | | |

## Unions

Use `|` to join two expressions.

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

## *More 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 <section> that directly contains h1#section-name
  //section[h1[@id='section-name']]
```

```
# Find a <section> that contains h1#section-name
# (Same as above, but use descendant-or-self instead of child)
  //section[//*[@id='section-name']]
```

```
# like jQuery's $().closest('.box')
  ./ancestor-or-self::[@class="box"]
```

```
# Find <item> and check its attributes
  //item[@price > 2*@discount]
```

## *References*

- Xpath test bed (whitebeam.org)