XPATH クイックリファレンス

軸、述語、関数、演算子、ノード選択

構文

パス式

`/`	ルートノード（絶対パスの開始）
`/bookstore/book`	直接の子要素を選択
`//book`	どこにでもある book ノードをすべて選択
`.`	現在のコンテキストノード
`..`	現在のノードの親
`@lang`	lang という名前の属性
`node()`	任意の種類のノード
`*`	任意の要素ノード
`@*`	任意の属性

基本的な例

```
/html/body/div          # absolute path to <div>
//input[@type='text']   # all text inputs
//div[@class='main']/*  # children of div.main
//a/@href               # all href attributes
```

パスの結合

```
//book/title | //book/price   # union of two paths
//h1 | //h2 | //h3           # multiple element types
```

軸

軸の方向

`child::`	直接の子（デフォルト軸）
`parent::`	直接の親
`ancestor::`	ルートまでのすべての祖先
`ancestor-or-self::`	祖先＋現在のノード
`descendant::`	すべての子孫
`descendant-or-self::`	子孫＋現在のノード
`following::`	文書内で現在のノードより後のすべてのノード
`following-sibling::`	現在より後の兄弟
`preceding::`	文書内で現在のノードより前のすべてのノード
`preceding-sibling::`	現在より前の兄弟
`self::`	現在のノードのみ
`attribute::`	現在のノードの属性
`namespace::`	名前空間ノード

軸の例

```
//div/child::p            # <p> children of <div>
//td/parent::tr           # <tr> parent of <td>
//h2/following-sibling::p  # <p> after <h2>
//li/ancestor::ul         # <ul> containing <li>
```

述語

述語によるフィルタリング

```
//book[1]                 # first book element
//book[last()]            # last book element
//book[position() < 3]    # first two books
//book[@lang='en']        # books with lang="en"
//book[price > 30]        # books with price > 30
```

述語パターン

`[n]`	位置 n の要素（1 始まり）
`[last()]`	最後の要素
`[last()-1]`	最後から 2 番目
`[@attr]`	属性を持つ
`[@attr='val']`	属性が値と一致
`[element]`	子要素を持つ
`[element='text']`	子要素がテキストを含む
`[not(@attr)]`	属性を持たない

連鎖述語

```
//div[@class='list']//a[1]        # first <a> in div.list
//input[@type='text'][@name='q']  # AND condition
//book[price>30][@lang='en']      # multiple conditions
```

関数

文字列関数

`contains(s, sub)`	s が sub を含む場合に真
`starts-with(s, pre)`	s が pre で始まる場合に真
`string-length(s)`	文字列の長さ
`normalize-space(s)`	空白をトリムして正規化
`concat(a, b, ...)`	文字列を結合
`substring(s, pos, len)`	部分文字列を抽出（1 始まり）
`translate(s, from, to)`	文字ごとに置換

数値関数

`sum(node-set)`	数値の合計
`count(node-set)`	ノード数
`floor(n)`	切り捨て
`ceiling(n)`	切り上げ
`round(n)`	最も近い整数に丸め
`number(val)`	数値に変換

関数の例

```
//div[contains(@class, 'active')]
//a[starts-with(@href, 'https')]
//p[string-length(text()) > 100]
//ul[count(li) > 5]
```

演算子

比較演算子

`=`	等しい
`!=`	等しくない
`<`	より小さい
`<=`	以下
`>`	より大きい
`>=`	以上

論理演算子と算術演算子

`and`	論理 AND
`or`	論理 OR
`not()`	論理 NOT（関数）
`+`	加算
`-`	減算
`*`	乗算
`div`	除算
`mod`	剰余
`|`	ノードセットの和集合

演算子の例

```
//book[price > 20 and price < 50]
//item[@type='a' or @type='b']
//span[not(contains(@class, 'hidden'))]
```

ノードテスト

ノードの種類

`node()`	任意のノード（要素、テキスト、コメント、PI）
`text()`	テキストノードのみ
`comment()`	コメントノードのみ
`processing-instruction()`	処理命令ノード
`*`	任意の要素ノード
`@*`	任意の属性ノード
`element-name`	特定の名前を持つ要素

ノードテストの例

```
//p/text()                # text content of <p>
//div/comment()           # comments inside <div>
//body/node()             # all child nodes of <body>
//div/*                   # all element children of <div>
```

ブール関数

`true()`	ブール値の真
`false()`	ブール値の偽
`boolean(expr)`	ブール値に変換
`not(expr)`	ブール値を否定
`lang(code)`	ノードの言語が一致する場合に真

省略形

短縮形と長縮形

`(none)`	child::（デフォルト軸）
`@`	attribute::
`//`	/descendant-or-self::node()/
`.`	self::node()
`..`	parent::node()
`[n]`	[position()=n]

省略形の例

```
# These pairs are equivalent:
child::div                →  div
attribute::href           →  @href
/descendant-or-self::node()/p  →  //p
self::node()              →  .
parent::node()            →  ..
```

よく使う省略パターン

```
//div[@id='main']       # div with id="main"
//table//td             # all <td> in any <table>
../sibling              # sibling via parent
.//span                 # span descendants of context
```

よく使うパターン

Web スクレイピング / テスト

```
//input[@name='username']          # form input by name
//button[text()='Submit']          # button by text
//div[contains(@class, 'error')]   # element by partial class
//a[contains(@href, 'login')]      # link by partial href
```

条件付き選択

```
//div[@class='item'][.//span[@class='price']]
//tr[td[1]='Active']               # row where 1st cell = Active
//*[contains(text(), 'Warning')]   # any element with text
```

Python での XPath（lxml）

```
from lxml import html
tree = html.fromstring(page_content)
links = tree.xpath('//a/@href')
titles = tree.xpath('//h2/text()')
```

Selenium での XPath

```
driver.find_element(By.XPATH, "//input[@id='search']")
driver.find_elements(By.XPATH, "//li[@class='result']")
```