XPather Cheatsheet

- XPather
- XPath
- RegExp

XPather 1.4 Overview

XPather v1.4 main features overview.

- DOMInspector navigation toolbar (XPath Toolbar)
- Customizable XPath generation (XPath toolbar menu)
- XPath evaluation
- XPath/Regexp syntax checking
- Feature rich XPath results Browser
- Selects XPath results in the DOMInspector
- Relative XPath support (Parent Toolbar)
- Namespaces and default namespaces
- Frames, IFrames support
- Cross-frame evaluation
- Optional RegExp content matching
- Content extraction tool (text, HTML, WebClip, ...)
- Accessible from DOM Inspector, or browser context menu
- This cheetsheet ;)

For complete documentation go the XPather web documentation!

Viktor Zigo http://xpath.alephzarro.com

XPath Overview

The XPather uses XPath engine provided by Gecko. XPath is a W3C standard.

XPath examples

//hr[@class='edge' and position()=1]	every first hr of 'edge' class
//table[count(tr)=1 and count(tr/td)=2]	all tables with 1 row and 2 cols
//div/form/parent::*	all divs that have form
./div/b	a relative path
//table[parent::div[@class="pad"] and not(@id)]//a	any anchor in a table without id, contained in a div of "pad" class
/html/body/div/*[preceding-sibling::h4]	give me whatever after h4
//tr/td[font[@class="head" and text()="TRACK"]]	all td that has font of a "head" class and text "TRACK"
./table/tr[last()]	the last row of a table
//rdf:Seq/rdf:li/em:id	using namespaces
//a/@href	

hrefs of all anchors

//*[count(*)=3]

all nodes with 3 children

//var|//acronym

all vars and acronyms

XPath functions

XPath functions.

Conversion:
`boolean( [object] )`
`string( [object] )`
`number( [object] )`
Math:
`ceiling( number )`
`floor( number )`
`round( decimal )`
`sum( node-set )`
Logic:
`true()`
`false()`
`not( expr )`
Node:
`lang(string)`
`name([node-set])`
`namespace-uri([node-set])`
`text()`
Context:
`count(node-set)`
`function-available( name )`
`last()`
`position()`
	String:
	`contains( haystack-string needle-string )`
	`concat( string1 string2 [ stringn]* )`
	`normalize-space( string )`
	`starts-with(haystack needle)`
	`string-length( [string] )`
	`substring(string start [ length])`
	`substring-after(haystack needle)`
	`substring-before(haystack needle)`
	`translate( string abc XYZ)`

XPath axes

Axes are relations two nodes. Each axis has also shortcut (in parenthesis).

XPath Axes.

• `ancestor`	• `following`
• `ancestor-or-self`	• `following-sibling`
• `attribute` (@)	• `parent`(..)
• `child` (/)	• `preceding`
• `descendant` (//)	• `preceding-sibling`
• `descendant-or-self`	• `self` (.)

Regular Expressions

More precisely: regular expresions in JavaScript, as the XPather requires regexps in JavaScript literal format, that is:

/.*/	matches everything
/ab+c/i	atches `abc`, `abbc`, `ABbc`

Processing modifiers.

g	global match
i	ignore case
m	match over multiple lines

Special characters in regular expressions.

*	0 or more
+	1 or many
?	0 or 1
{*n*}	exactly *n*
{*n*,}	*n* or more
{*n*,*m*}	*n* to *m*
\	meta character for special chars, or "take literally" otherwise
^	begining; or negation in character set
$	match end; EOL if multiline
.	any char
(*x*)	group; match & capture *x*
(?:*x*)	group; match *x*(dont capture)
x(?=*y*)	match *x* if followed by *y*
x(?!*y*)	match *x* if not followed by *y*
x|*y*	*x* or *y*
[*xyz*]	character set, match any char from *xyz*
[^*xyz*]	complemented character set
[\b]	backspace
\b \B	word boundary
\c*X*	control character *X* in a string
\d \D	digit = [0-9]; not digit
\f	form feed
\n	linefeed
\r	carriage return,
\s \S	single whitespace = [\f\n\r\t\v\u00A0\u2028\u2029]
\t	tab
\v	vertical tab
\w \W	word char=alphanumeric char+_= [A-Za-z0-9_]
\n	group back reference (number, left count)
\0	null
\x*hh* \u*hhhh*	hex code char

Viktor Zigo, http://xpath.alephzarro.com , Mar 16 2008