XPath • CSS • DOM • Selenium Rosetta Stone and Cookbook

Sprinkled with Selenium usage tips, this is both a general-purpose set of recipes for each technology as well as a cross-reference to map from one to another.
The validation suite for this reference chart (http://bit.ly/gTd5oc) provides example usage for each recipe supported by Selenium (the majority of them).

Category	Recipe	XPath (1.0 – 2.0)	CSS (CSS1 – 3)	DOM	Selenium
General	Whole web page	xpath=/html	css=html	document.documentElement	*NA*
	Whole web page body	xpath=/html/body	css=body	document.body	*NA*
	All text nodes of web page	//text() ⌦	*NA*	*NA*	*NA*
	Element <E> by absolute reference	xpath=/html/body/.../.../.../E	css=body>...>...>...>E	document.body.childNodes[i]...childNodes[j]	*NA*
Tag	Element <E> by relative reference	//E	css=E	document.gEBTN('E')[0]	*NA*
	Second <E> element anywhere on page	xpath=(//E)[2]	*NA*	document.gEBTN('E')[1]	*NA*
	Image element	//img	css=img	document.images[0]	*NA*
	Element <E> with attribute A	//E[@A]	css=E[A]	dom=for each (e in document.gEBTN('E')) if (e.A) e[1]	*NA*
	Element <E> with attribute A containing text 't' exactly	//E[@A='t']	css=E[A='t'] [2]	*NA*	*NA*
	Element <E> with attribute A containing text 't'	//E[contains(@A,'t')]	**css=E[A*='t']** [2]	*NA*	*NA*
	Element <E> whose attribute A begins with 't'	//E[starts-with(@A, 't')]	**css=E[A^='t']** [2]	*NA*	*NA*
	Element <E> whose attribute A ends with 't'	**//E[ends-with(@A, 't')]** ⌦ ◀OR▶ //E[substring(@A, string-length(@A) - string-length('t')+1)='t']	**css=E[A$='t']** [2]	*NA*	*NA*
	Element <E> with attribute A containing word 'w'	//E[contains(concat('◉', @A, '◉'), '◉w◉')]	**css=E[A~='w']** [2]	*NA*	*NA*
	Element <E> with attribute A matching regex 'r'	**//E[matches(@A, 'r')]** ⌦	*NA*	*NA*	*NA*
	Element <E1> with id I1 or element <E2> with id I2	//E1[@id=I1] | //E2[@id=I2]	css=E1#I1,E2#I2	*NA*	*NA*
	Element <E1> with id I1 or id I2	//E1[@id=I1 or @id=I2]	css=E1#I1,E1#I2	*NA*	*NA*
Attribute[3]	Attribute A of element <E>	//E/@A ⌦ {Se: //E@A }	*NA* {Se: css=E@A }	document.gEBTN('E')[0].getAttribute('A') ⌦ {Se: document.gEBTN('E')[0]@A }	*NA*
	Attribute A of any element	//*/@A ⌦ {Se: //*@A }	*NA* {Se: css=*@A }	*NA*	*NA*
	Attribute A1 of element <E> where attribute A2 is 't' exactly	//E[@A2='t']/@A1 ⌦ {Se: //E[@A2='t']@A1 }	*NA* {Se: css=E[A2='t']@A1 }	*NA*	*NA*
	Attribute A of element <E> where A contains 't'	//E[contains(@A,'t')]/@A ⌦ {Se: //E[contains(@A,'t')]@A }	*NA* {Se: **css=E[A*='t']**@A }	*NA*	*NA*
Id & Name	Element <E> with id I	//E[@id='I']	css=E#I	*NA*	*NA*
	Element with id I	//*[@id='I']	css=#I	document.gEBI('I')	id=I
	Element <E> with name N	//E[@name='N']	css=E[name=N]	*NA*	*NA*
	Element with name N	//*[@name='N']	css=[name=N]	document.getElementsByName('N')[0]	name=N
	Element with id X or, failing that, a name X	//*[@id='X' or @name='X']	*NA*	*NA*	X ◀OR▶ identifier=X
	Element with name N & specified 0-based index 'v'	//*[@name='N'][v+1]	css=[name=N]:nth-child(v+1)	*NA*	name=N index=v
	Element with name N & specified value 'v'	//*[@name='N'][@value='v']	css=[name=N][value='v']	*NA*	name=N value=v
Lang & Class	Element <E> is explicitly in language L or subcode	//E[@lang='L' or starts-with(@lang, concat('L', '-'))]	css=E[lang|=L]	NA	*NA*
	Element <E> is in language L or subcode (possibly inherited)	NA	css=E:lang(L)	NA	*NA*
	Element with a class C	//*[contains(concat('◉', @class, '◉'), '◉C◉')]	css=.C	document.getElementsByClassName('C')[0]	*NA*
	Element <E> with a class C	//E[contains(concat('◉', @class, '◉'), '◉C◉')]	css=E.C	*NA*	*NA*
Text & Link	Element containing text 't' exactly	//*[.='t']	*NA*	*NA*	*NA*
	Element <E> containing text 't'	//E[contains(text(),'t')]	css=E:contains('t') [4]	*NA*	*NA*
	Link element	//a	css=a	document.links[0]	*NA*
	<a> containing text 't' exactly	//a[.='t']	*NA*	*NA*	link=t
	<a> containing text 't'	//a[contains(text(),'t')]	css=a:contains('t') [4]	*NA*	*NA*
	<a> with target link 'url'	//a[@href='url']	css=a[href='url']	*NA*	*NA*
	Link URL labeled with text 't' exactly	//a[.='t']/@href	*NA*	*NA*	*NA*
Parent & Child	First child of element <E>	//E/*[1]	css=E > *:first-child { Se: css=E > * }	document.gEBTN('E')[0].firstChild[5]	*NA*
	First <E> child	//E[1]	**css=E:first-of-type** ⌦ { Se: css=E }	document.getEBTN('E')[0]	*NA*
	Last child of element E	//E/*[last()]	**css=E *:last-child**	document.gEBTN('E')[0].lastChild[5]	*NA*
	Last <E> child	//E[last()]	**css=E:last-of-type** ⌦	document.gEBTN(E)[document.gEBTN(E).length-1]	*NA*
	Second <E> child	//E[2] ◀OR▶ //E/following-sibling::E	**css=E:nth-of-type(2)** ⌦	document.getEBTN('E')[1]	*NA*
	Second child that is an <E> element	//*[2][name()='E']	**css=E:nth-child(2)**	*NA*	*NA*
	Second-to-last <E> child	//E[last()-1]	**css=E:nth-last-of-type(2)** ⌦	document.gEBTN(E)[document.gEBTN(E).length-2]	*NA*
	Second-to-last child that is an <E> element	//*[last()-1][name()='E']	**css=E:nth-last-child(2)** ⌦	*NA*	*NA*
	Element <E1> with only <E2> children	//E1/[E2 and not(*[not(self::E2)])]	*NA*	*NA*	*NA*
	Parent of element <E>	//E/..	*NA*	document.gEBTN('E')[0].parentNode	*NA*
	Descendant <E> of element with id I using specific path	//*[@id='I']/ . . ./. . ./. . ./E	css=#I > . . . > . . . > . . . > E	document.gEBI('I')...gEBTN('E')[0]	*NA*
	Descendant <E> of element with id I using unspecified path	//*[@id='I']//E	css=#I E	document.gEBI('I').gEBTN('E')[0]	*NA*
	Element <E> with no children	//E[count(*)=0]	**css=E:empty**	*NA*	*NA*
	Element <E> with an only child	//E[count(*)=1]	*NA*	*NA*	*NA*
	Element <E> that *is* an only child	//E[count(preceding-sibling::*)+count(following-sibling::*)=0]	**css=E:only-child**	*NA*	*NA*
	Element <E> with no <E> siblings	//E[count(../E) = 1]	**css=E:only-of-type** ⌦	*NA*	*NA*
	Every Nth element starting with the (M+1)th	//E[position() mod N = M + 1]	**css=E:nth-child(Nn + M)**	*NA*	*NA*
Sibling	Element <E1> following some sibling <E2>	//E2/following-sibling::E1	**css=E2 ~ E1**	*NA*	*NA*
	Element <E1> immediately following sibling <E2>	//E2/following-sibling::*[1][name()='E1']	css=E2 + E1	*NA*	*NA*
	Element <E1> following sibling <E2> with one intermediary	//E2/following-sibling::*[2][name()='E1']	css=E2 + * + E1	*NA*	*NA*
	Sibling element immediately following <E>	//E/following-sibling::*	css=E + *	document.gEBTN('E')[0].nextSibling[5]	*NA*
	Element <E1> preceding some sibling <E2>	//E2/preceding-sibling::E1	*NA*	*NA*	*NA*
	Element <E1> immediately preceding sibling <E2>	//E2/preceding-sibling::*[1][name()='E1']	*NA*	*NA*	*NA*
	Element <E1> preceding sibling <E2> with one intermediary	//E2/preceding-sibling::*[2][name()='E1']	*NA*	*NA*	*NA*
	Sibling element immediately preceding <E>	//E/preceding-sibling::*[1]	*NA*	document.gEBTN('E2')[0].previousSibling[5]	*NA*
Table Cell	Cell by row and column (e.g. 3rd row, 2nd column)	//*[@id='TestTable']//tr[3]//td[2] {Se: //*[@id='TestTable'].2.1 }	**css=#TestTable tr:nth-child(3) td:nth-child(2)** {Se: css=#TestTable.2.1 }	document.gEBI('TestTable').gEBTN('tr')[2].gEBTN('td')[1] {Se: document.gEBI('TestTable').2.1 }	*NA*
	Cell immediately following cell containing 't' exactly	//td[preceding-sibling::td='t']	*NA*	*NA*	*NA*
	Cell immediately following cell containing 't'	//td[preceding-sibling::td[contains(.,'t')]]	**css=td:contains('t') ~ td** [4]	*NA*	*NA*
Dynamic	User interface element <E> that is disabled	//E[@disabled]	**css=E:disabled**	*NA*	*NA*
	User interface element that is enabled	//*[not(@disabled)]	**css=*:enabled**	*NA*	*NA*
	Checkbox (or radio button) that is checked	//*[@checked]	**css=*:checked**	*NA*	*NA*
	Element being designated by a pointing device	*NA*	css=E:hover ⌦	*NA*	*NA*
	Element has keyboard input focus	*NA*	css=E:focus ⌦	*NA*	*NA*
	Unvisited link	*NA*	css=E:link ⌦	*NA*	*NA*
	Visited link	*NA*	css=E:visited ⌦	*NA*	*NA*
	Active element	*NA*	css=E:active ⌦	*NA*	*NA*

LEGEND

▨	XPath
🟩	CSS
🟦	DOM
🟧	Selenium
{Se: . . . }	Selenium-only variation
⌦	Not supported by Selenium
◉	Space character
expression	CSS3 or XPath 2.0

DOM abbreviations:

gEBI	getElementById
gEBTN	getElementsByTagName

Download the latest version from Simple-Talk http://bit.ly/gTd5oc.

General Notes

Indexing (all): XPath and CSS use 1-based indexing; DOM and Selenium's table syntax use 0-based indexing.

Prefixes (all): xpath= required unless expression starts with // ● dom= required unless expression starts with "document." ● css= ***always*** required ● identifier= ***never*** required.

Cardinality (Selenium): XPath and CSS may specify a node set **or** a single node; DOM **must** specify a single node. When a node set is specified, Selenium returns just the first node.

Content (XPath): Generally should use normalize-space() when operating on display text.

Footnotes

[1] DOM has limited capability with a simple 'document…' expression; however, arbitrary JavaScript code may be used as shown in this example.

[2] CSS does not support qualifying elements with the **style** attribute, as in div[style*='border-width'].

[3] Selenium uses a special syntax for returning attributes; normal XPath, CSS, and DOM syntax will fail.

[4] CSS: The CSS2 **contains** function is ***not in CSS3***; however, Selenium supports the superset of CSS1, 2, and 3.

[5] DOM: **firstChild**, **lastChild**, **nextSibling**, and **previousSibling** are problematic with mixed content; they will point to empty text nodes rather than desired elements depending on whitespace in web page

XPath • CSS • DOM • Selenium
Rosetta Stone and Cookbook

Sprinkled with Selenium usage tips, this is both a general-purpose set of recipes for each technology as well as a cross-reference to map from one to another. The validation suite for this reference chart (http://bit.ly/gTd5oc) provides example usage for each recipe supported by Selenium (the majority of them).

General

Whole web page
xpath=/html
css=html
document.documentElement

Whole web page body
xpath=/html/body
css=body
document.body

All text nodes of web page
//text() ⊠

Element <E> by absolute reference
xpath=/html/body/.../.../.../E
css=body>...>...>...>E
document.body.childNodes[i]...childNodes[j]

Tag

Element <E> by relative reference
//E
css=E
document.gEBTN('E')[0]

Second <E> element anywhere on page
xpath=(//E)[2]
document.gEBTN('E')[1]

Image element
//img
css=img
document.images[0]

Element <E> with attribute A
//E[@A]
css=E[A]
dom=for each (e in document.gEBTN('E')) if (e.A) e [1]

Element <E> with attribute A containing text 't' exactly
//E[@A='t']
css=E[A='t'] [2]

Element <E> with attribute A containing text 't'
//E[contains(@A,'t')]
css=E[A*='t'] [2]

Element <E> whose attribute A begins with 't'
//E[starts-with(@A, 't')]
css=E[A^='t'] [2]

Element <E> whose attribute A ends with 't'
//E[ends-with(@A, 't')] ⊠ ◀OR▶
//E[substring(@A, string-length(@A) - string-length('t')+1)='t']
css=E[A$='t'] [2]

Element <E> with attribute A containing word 'w'
//E[contains(concat('◉', @A, '◉'), '◉w◉')
css=E[A~='w'] [2]

Element <E> with attribute A matching regex 'r'
//E[matches(@A, 'r')] ⊠

Element <E1> with id I1 or element <E2> with id I2
//E1[@id=I1] | //E2[@id=I2]
css=E1#I1,E2#I2

Element <E1> with id I1 or id I2
//E1[@id=I1 or @id=I2]
css=E1#I1,E1#I2

Attribute [3]

Attribute A of element <E>
//E/@A ⊠ {Se: //E@A }
{Se: css=E@A }
document.gEBTN('E')[0].getAttribute('A') ⊠
{Se: document.gEBTN('E')[0]@A }

Attribute A of any element
//*/@A ⊠ {Se: //*@A }
{Se: css=*@A }

Attribute A1 of element <E> where attribute A2 is 't' exactly
//E[@A2='t']/@A1 ⊠ {Se: //E[@A2='t']@A1 }
{Se: css=E[A2='t']@A1 }

Attribute A of element <E> where A contains 't'
//E[contains(@A,'t')]/@A ⊠ {Se: //E[contains(@A,'t')]@A }
{Se: **css=E[A*='t']**@A }

Id & Name

Element <E> with id I
//E[@id='I']
css=E#I

Element with id I
//*[@id='I']
css=#I
document.gEBI('I')
id=I

Element <E> with name N
//E[@name='N']
css=E[name=N]

Element with name N
//*[@name='N']
css=[name=N]
document.getElementsByName('N')[0]
name=N

Element with id X or, failing that, a name X
//*[@id='X' or @name='X']
X ◀OR▶ identifier=X

Element with name N & specified 0-based index 'v'
//*[@name='N'][v+1]
css=[name=N]:nth-child(v+1)
name=N index=v

Element with name N & specified value 'v'
//*[@name='N'][@value='v']
css=[name=N][value='v']
name=N value=v

Lang & Class

Element <E> is explicitly in language L or subcode
//E[@lang='L' or starts-with(@lang, concat('L', '-'))]
css=E[lang|=L]

Element <E> is in language L or subcode (possibly inherited)
css=E:lang(L)

Element with a class C
//*[contains(concat('◉', @class, '◉'), '◉C◉')]
css=.C
document.getElementsByClassName('C')[0]

Element <E> with a class C
//E[contains(concat('◉', @class, '◉'), '◉C◉')]
css=E.C

Text & Link

Element containing text 't' exactly
//*[.='t']

Element <E> containing text 't'
//E[contains(text(),'t')]
css=E:contains('t') [4]

Link element
//a
css=a
document.links[0]

<a> containing text 't' exactly
//a[.='t']
link=t

<a> containing text 't'
//a[contains(text(),'t')]
css=a:contains('t') [4]

<a> with target link 'url'
//a[@href='url']
css=a[href='url']

Link URL labeled with text 't' exactly
//a[.='t']/@href

Parent & Child

First child of element <E>
//E/*[1]
css=E > *:first-child {Se: css=E > * }
document.gEBTN('E')[0].firstChild [5]

First <E> child
//E[1]
css=E:first-of-type ⊠ {Se: css=E }
document.getEBTN('E')[0]

Last child of element E
//E/*[last()]
css=E *:last-child
document.gEBTN('E')[0].lastChild [5]

Last <E> child
//E[last()]
css=E:last-of-type ⊠
document.gEBTN(E)[document.gEBTN(E).length-1]

Second <E> child
//E[2] ◀OR▶ //E/following-sibling::E
css=E:nth-of-type(2) ⊠
document.getEBTN('E')[1]

Second child that is an <E> element
//*[2][name()='E']
css=E:nth-child(2)

Second-to-last <E> child
//E[last()-1]
css=E:nth-last-of-type(2) ⊠
document.gEBTN(E)[document.gEBTN(E).length-2]

Second-to-last child that is an <E> element
//*[last()-1][name()='E']
css=E:nth-last-child(2) ⊠

Element <E1> with only <E2> children
//E1/[E2 and not(*[not(self::E2)])]

Parent of element <E>
//E/..
document.gEBTN('E')[0].parentNode

Descendant <E> of element with id I using specific path
//*[@id='I']/ . . ./. . ./. . ./E
css=#I > . . . > . . . > . . . > E
document.gEBI('I')...gEBTN('E')[0]

Descendant <E> of element with id I using unspecified path
//*[@id='I']//E
css=#I E
document.gEBI('I').gEBTN('E')[0]

Element <E> with no children
//E[count(*)=0]
css=E:empty

Element <E> with an only child
//E[count(*)=1]

Element <E> that is an only child
//E[count(preceding-sibling::*)+count(following-sibling::*)=0]
css=E:only-child

Element <E> with no <E> siblings
//E[count(../E) = 1]
css=E:only-of-type ⊠

Every Nth element starting with the (M+1)th
//E[position() mod N = M + 1]
css=E:nth-child(Nn + M)

Sibling

Element <E1> following some sibling <E2>
//E2/following-sibling::E1
css=E2 ~ E1

Element <E1> immediately following sibling <E2>
//E2/following-sibling::*[1][name()='E1']
css=E2 + E1

Element <E1> following sibling <E2> with one intermediary
//E2/following-sibling::*[2][name()='E1']
css=E2 + * + E1

Sibling element immediately following <E>
//E/following-sibling::*
css=E + *
document.gEBTN('E')[0].nextSibling [5]

Element <E1> preceding some sibling <E2>
//E2/preceding-sibling::E1

Element <E1> immediately preceding sibling <E2>
//E2/preceding-sibling::*[1][name()='E1']

Element <E1> preceding sibling <E2> with one intermediary
//E2/preceding-sibling::*[2][name()='E1']

Sibling element immediately preceding <E>
//E/preceding-sibling::*[1]
document.gEBTN('E2')[0].previousSibling [5]

Table Cell

Cell by row and column (e.g. 3rd row, 2nd column)
//*[@id='TestTable']//tr[3]//td[2]
{Se: //*[@id='TestTable'].2.1 }
css=#TestTable tr:nth-child(3) td:nth-child(2)
{Se: css=#TestTable.2.1 }
document.gEBI('TestTable').gEBTN('tr')[2].gEBTN('td')[1]
{Se: document.gEBI('TestTable').2.1 }

Cell immediately following cell containing 't' exactly
//td[preceding-sibling::td='t']

Cell immediately following cell containing 't'
//td[preceding-sibling::td[contains(.,'t')]]
css=td:contains('t') ~ td [4]

Dynamic

User interface element <E> that is disabled
//E[@disabled]
css=E:disabled

User interface element that is enabled
//*[not(@disabled)]
css=*:enabled

Checkbox (or radio button) that is checked
//*[@checked]
css=*:checked

Element being designated by a pointing device
css=E:hover ⊠

Element has keyboard input focus
css=E:focus ⊠

Unvisited link
css=E:link ⊠

Visited link
css=E:visited ⊠

Active element
css=E:active ⊠

Footnotes

[1] DOM has limited capability with a simple 'document...' expression; however, arbitrary JavaScript code may be used as shown in this example.
[2] CSS does not support qualifying elements with the **style** attribute, as in div[style*='border-width'].
[3] Selenium uses a special syntax for returning attributes; normal XPath, CSS, and DOM syntax will fail.
[4] CSS: The CSS2 **contains** function is ***not in CSS3***; however, Selenium supports the superset of CSS1, 2, and 3.
[5] DOM: **firstChild**, **lastChild**, **nextSibling**, and **previousSibling** are problematic with mixed content; they will point to empty text nodes rather than desired elements depending on whitespace in web page source.

{Se: . . . }	Selenium-only variation
⊠	Not supported by Selenium
◉	Space character
expression	CSS3 or XPath 2.0

DOM abbreviations:

gEBI	getElementById
gEBTN	getElementsByTagName

General Notes

Indexing (all): XPath and CSS use 1-based indexing; DOM and Selenium's table syntax use 0-based indexing.
Prefixes (all): xpath= required unless expression starts with // • dom= required unless expression starts with "document." • css= ***always*** required • identifier= ***never*** required.
Cardinality (Selenium): XPath and CSS may specify a node set **or** a single node; DOM **must** specify a single node. When a node set is specified, Selenium returns just the first node.
Content (XPath): Generally should use normalize-space() when operating on display text.

Key ▨ XPath 🟩 CSS 🟦 DOM 🟧 Selenium